UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

TREBALL FINAL DE GRAU

Titulació:
Enginyeria de Sistemes Audiovisuals.

Alumne:
Bernat Vilella Fite

Enunciat TFG:
Creació d'una aplicació per a telèfons mòbils Android per a la creació i edició d'imatges fotogràfiques.

Director/a del TFG:
Jordi Marco Gomez

Convocatòria de lliurament del TFG:
Tardor 2018.

Resum

Aquest projecte de final de grau consisteix en el desenvolupament d'una aplicació d'Android per a l'edició d'imatges digitals. En l'edició d'aquestes s'intenta utilitzar codi propi per tal d'aprendre el funcionament de cada efecte que es vol aplicar a les imatges i s'investiga sobre el desenvolupament d'aplicacions per a smartphones Android.

Resumen

Este proyecto de fin de grado consiste en el desarrollo de una aplicación de Android para la edición de imágenes digitales. En la edición de estas se intenta utilizar código propio para aprender el funcionamiento de cada efecto que se quiere aplicar a las imágenes y se investiga sobre el desarrollo de aplicaciones para smartphones Android.

Abstract

This end-of-degree project consists in the development of an Android application for digital image edition. For this, we try to use our own code in order to learn the operation of each effect that is applied to the images and investigate the development of applications for Android smartphones.

Agraïments

Primerament vull agrair al tutor d'aquest projecte, Jordi Marco, tant l'acceptació d'aquesta proposta com l'aportació d'idees a la proposta inicial del projecte, i durant el desenvolupament d'aquest. També l'ajuda proporcionada durant el redactat de la memòria.

Seguidament, donar les gràcies també als professors que m'han impartit classe durant aquests últims 4 anys i m'han format fins al punt de fer-me adonar de les meves capacitats, que m'han ajudat i han aportat el seu gra d'arena en aquest projecte.

També fer especial menció a la meva companya de classe Cristina Uroz, ja que aquest idea va néixer de l'evolució d'un projecte anterior realitzat en una assignatura de programació.

A la família, pel suport donat en els temps difícils i poder gaudir junts del resultat d'aquest projecte amb ells, i poder-ne realitzar molts més junts, i als amics que han estat explotats com a testers per un preu accessible.

A tots, moltes gràcies per ser-hi.

Index

1 Introducció

1.1 Motivació

La fotografia com l'entenem avui en dia, va néixer l'any 1826 de la mà de Niepce, un científic francès obsessionat en preservar per a la posteritat les vistes des de la finestra de casa seva. I ho va aconseguir, ja que va ser el primer en aconseguir una imatge fixa d'un paisatge i que a més aquesta no desapareixes amb el pas del temps.
En aquells temps la fotografia era un luxe que només es podia permetre l'alta societat, ja que requeria elements estranys i cars com la plata, que es requeria per a crear els daguerreotips, un tipus de fotografies famoses entre l'alta societat durant la segona meitat del segle XIX. És per això que la captació d'imatges fixes resultava cara, tant en termes econòmics a causa dels materials utilitzats, com en termes temporal, ja que una simple captura podia durar bastant temps, inclús diverses hores. Això no suposa cap problema si es retratava un paisatge, però si es tracta d'un retrat familiar el fet d'estar 3 hores sense moure's ni canviar l'expressió facial resulta una tasca complicada.

Amb el pas dels anys la tecnologia ha evolucionat a nivells impensables. Hem passat de fabricar rodes de pedra a enviar un humà a la Lluna en aproximadament uns 5 mil anys i la majoria d'estris que coneixem actualment són evolucions d'estris anteriors. Això també ha passat amb la fotografia. Les cares i aparatoses càmeres han evolucionat fins a tal punt que les podem portar a la butxaca i fins i tot a vegades les perdem! Són els smartphones. Pràcticament avui en dia tothom té un smartphone amb càmera integrada, i se'n va a la feina i de camí veu l'últim model del cotxe dels seus somnis, va a dinar i li encanta la presentació del seu plat, arriba a casa a la nit i el sorprèn l'adorabilitat del seu gat dormint. I tots sabem que passa avui en dia en aquesta situació; CLICK! Imatge guardada! Suposant que això ho féssim únicament mil milions de persones, una setena part de la població mundial, obtindríem una quantitat d'imatges bestial. Amb això ens adonem que les imatges s'han convertit en un dels elements més important del nostre dia a dia.

I perquè fem servir aquestes imatges? Per mostrar la nostra vida als amics, per expressar-nos artísticament, per comunicar-nos, però a vegades la imatge que hem captat no reflecteix al 100% el que volem mostrar. Transmet la idea principal però no ens convenç de forma completa, ja que hi ha algun petit detall minúscul que no ens agrada. I és aquí quan pensem en editar-la.

La gran majoria d'imatges que veiem avui en dia estan editades. A la televisió, a les revistes, als cartells publicitaris, a Internet, i això és perquè algú ha cregut oportú que la imatge on s'anuncia un refresc energètic tingui uns colors més vius, o que el blau de l'aigua de la piscina d'aquell bloc de pisos en venda no és prou clar, per exemple. Això es fa per tal de crear sensacions a les persones que veuen les imatges i els transmeti una idea determinada.

És d'aquí d'on sorgeix la idea de crear un editor d'imatges, ja que poder reflectir bé el que volem no sempre s'aconsegueix a la primera, i si s'afegeix a més el fet que anteriorment es deia, que pràcticament tothom avui en dia té un smartphone amb càmera integrada, la idea principal es transforma en la creació d'una aplicació per a smartphones que sigui capaç d'editar les imatges que generem les persones per tal de poder expressar les nostres idees.

Com a Projecte Final de Grau de la titulació d'Enginyeria de Sistemes Audiovisuals que ha tingut lloc a l'Escola Superior d'Enginyers Industrial, Aeroespacial i Audiovisual de Terrassa durant els anys 2014 i 2018 he volgut desenvolupar una app per a smartphones Android capaç de crear i editar imatges tant d'arxiu com captades amb la càmera de dispositiu.

La idea principal consistia a crear un software d'edició d'imatges per a ordinador, ja que la fotografia i el vídeo han estat una de les meves grans passions des de petit. Al cap d'un temps, va sorgir l'oportunitat de fer una assignatura de creació d'aplicacions Android, en la qual vaig veure el relativament senzill i extremadament divertit que era crear aplicacions per a poder realitzar algunes tasques quotidianes, com per exemple realitzar una llista de la compra o un simple conversor d'unitats, entre altres. El fet de ser aplicacions per a smartphone permet utilitzar-les en qualsevol situació, i això va ser el punt clau per a decidir que seria una aplicació per a smartphone i no per a ordinador.

1.2 Objectius

Com a objectiu principal, es vol desenvolupar una app per a dispositius Android funcional que tingui un nivell de qualitat acceptable i que pugui ser accessible des de la Play Store, el repositori d'apps oficial de Google, de manera gratuïta. Com a altres objectius podem destacar que es vol conèixer quins són els tractaments més usuals en el món de l'edició d'imatges i com es realitzen a nivell de píxel, així com investigar si les solucions trobades es poden optimitzar d'alguna manera o trobar-ne de més eficients. També es vol aconseguir dominar l'entorn de programació Android Studio en un nivell avançat i conèixer quines funcionalitats del llenguatge Java són necessàries per a crear aplicacions d'una complexitat elevada. Es pretén també conèixer quines necessitats tenen els usuaris quan volen editar una imatge per a pensar possibles millores *d'Interfície Gràfica d'Usuari* (GUI, acrònim en anglès de Graphic User Interface) que podrien estar disponibles en futures actualitzacions de l'app. Per últim es pretén agafar pràctica a l'hora de desenvolupar aplicacions per a mòbil, tant aprenent quines fases i tasques té un projecte d'aquestes característiques i en quin ordre succeeixen, per a preparar-se per al futur mercat laboral dins d'aquest món.

1.3 Estructura de la memòria

Per començar, i a continuació d'aquesta secció, trobareu una secció en la qual s'explica alguns conceptes necessaris per a poder desenvolupar l'aplicació. Aquests són el processament d'imatges i els fonaments bàsics sobre les aplicacions d'Android. En la secció de processament s'explica que és una imatge a nivell tècnic, quines característiques tenen i altres dades per tal de posar en situació a la gent que no té coneixements sobre tractament d'imatges. En la secció d'aplicacions Android es farà una descripció, tant del software utilitzat, que en aquesta ocasió ha estat Android Studio, com dels elements natius bàsics d'Android.

Es procedirà realitzant un anàlisi de mercat de les aplicacions que implementen funcionalitats semblants a la que volem desenvolupar i farà una comparativa entre elles.
Seguidament hi haurà l'apartat on s'explicara tot el desenvolupament de l'aplicació. Aquí es descriurà que s'ha volgut crear i com en l'apartat de disseny de l'aplicació, quines eines s'ha utilitzat i quina estructura té el nostre projecte d'Android, fent referència a les classes que s'ha creat, activitats i recursos.

A continuació es mostra el resultat de tot aquest desenvolupament i es finalitzarà exposant les conclusions i opinions que s'ha tingut durant la realització del projecte.

2 Background

En aquest apartat s'explica algunes nocions bàsiques sobre el processament d'imatges i el desenvolupament d'aplicacions Android. És una secció que pretén ensenyar com funcionen de manera senzilla els efectes que es vol implementar amb l'ajuda d'imatges de mostra per entendre millor el funcionament. També es mencionen els elements més fonamentals del programari utilitzat, que en aquesta ocasió és Android Studio.

2.1 Processament d'imatges

Una imatge és una representació visual que pretén simular el que es veu a través del sistema visual humà i amb la missió de transmetre informació. Aquestes representacions es basen en la llum, que es pot descompondre en tres colors primaris. Aquestes són el vermell, el verd i el blau, i reben el nom de components de color. De les inicials d'aquests colors en anglès s'obté el nom RGB.

Des del punt de vista informàtic, una imatge és un arxiu que conté informació visual. Sempre tenen forma rectangular i estan formades per pixels, fet que permet emmagatzemar aquesta informació en matrius de dades. Normalment s'utilitzen 3 bytes per a emmagatzemar aquestes dades. Cadascun d'aquests tres bytes emmagatzemen la informació de cada component de color, és a dir, un byte per a cada component, fet que comporta que el valor de cada component vagi desde el 0 al 255. Aquestes matrius poden contenir diferent tipus d'informació depenent de la codificació de la imatge. Bàsicament hi ha dos tipus diferents d'arxius:

- Imatges de color veritable: l'arxiu conté una matriu a la qual es guarda en cada posició d'aquesta la informació de cada pixel de la imatge. En cadascuna d'aquestes posicions s'emmagatzema una terna de bytes del tipus (R,G,B) en la qual el primer terme es correspon al valor de la component de color vermell d'aquell pixel, el segon al de color verd i el tercer al de color blau.
- Imatges indexades: en realitza un anàlisi de la imatge original i s'obté quines són les **n** combinacions de les tres components de color amb més probabilitat d'aparició. S'emmagatzemen en una taula aquestes combinacions, i en una matriu, per a cada pixel, s'indica l'índex de la posició de la taula de combinacions de colors del color més semblant al color real del pixel.

Per a la transmissió d'imatges, històricament s'ha fet servir dos mètodes. El primer és creant senyals directament amb les components RGB anteriorment comentades. El segon és utilitzant un senyal de luminància i dues de crominància, creades a partir de les components RGB. Aquests dos senyals de crominància, reben el nom de R-Y i B-Y, mentre que la luminància s'anomena Y. L'obtenció d'aquestes dos senyals es du a terme utilitzant les següents fórmules:

$$Y = (30 * R + 59 * G + 11 * B) / 100$$
$$R\text{-}Y = (70 * R - 59 * G - 11 * B) / 100$$
$$B\text{-}Y = (-30 * R - 59 * G + 89 * B) / 100$$

Aquests valors representen el percentatge amb el qual la vista humana dona més importància a cadascun dels components de color.

Per tal de poder aplicar els efectes d'una forma eficient, en els casos en què això sigui possible, s'investiga si és millor aplicar l'efecte sobre les components RGB o sobre els senyals de luminància i la crominància. En el cas d'alguns efectes és més fàcil treballar directament amb els valors dels canals RGB de cada pixel de la imatge, mentre que en altres casos només és possible aplicar les modificacions necessàries sobre la luminància i la crominància.

Per acabar i entrar en la part de l'explicació sobre com modifica la imatge cadascun dels efectes que es vol crear, s'introdueix el concepte de canal alfa. Aquí és on es guarda la informació sobre el nivell d'opacitat d'un pixel. No s'entrarà gaire en detall sobre aquest concepte, ja que només necessitarem aquest element en un sol efecte.

Totes les imatges que es mostren a continuació han estat modificades amb l'aplicació.

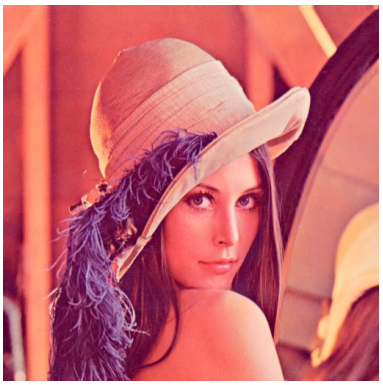
Balanç de blancs

També conegut com a temperatura de color o calidesa, es modifica variant les diferències de color R-Y i B-Y de manera inversa. També es pot fer aplicant la mateixa norma sobre els canals R i B únicament.

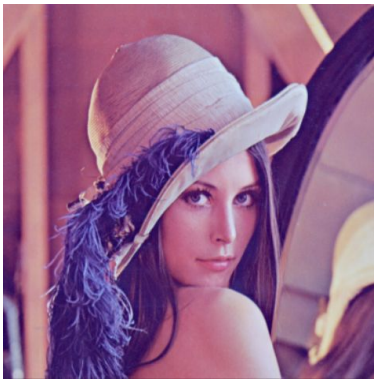
Mètode 1:
R-Y' = R-Y +- valor
B-Y' = B-Y -+ valor

Mètode 2:
R' = R +- valor
G' = G
B' = B -+ valor

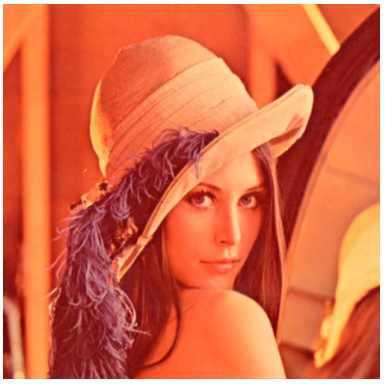
El resultat es reflecteix en què la imatge resultant és més freda o càlida:



Original.



Temperatura baixa.



Temperatura alta.

Blanc i negre

Es converteix una imatge que pot estar en color o no a escala de grisos. S'aconsegueix igualant cadascun dels tres valors RGB de cada pixel de la imatge al valor de la luminància d'aquest. [1]

$$R' = Y$$
$$G' = Y$$
$$B' = Y$$

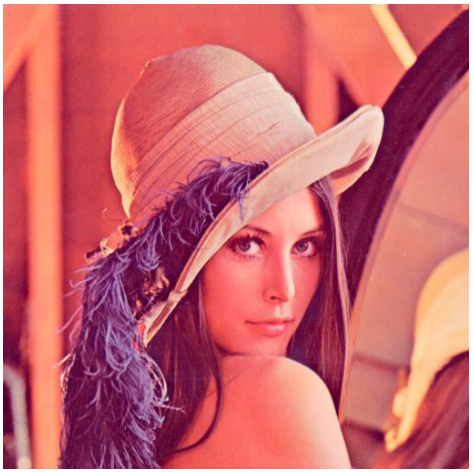
Original.

Blanc i negre.

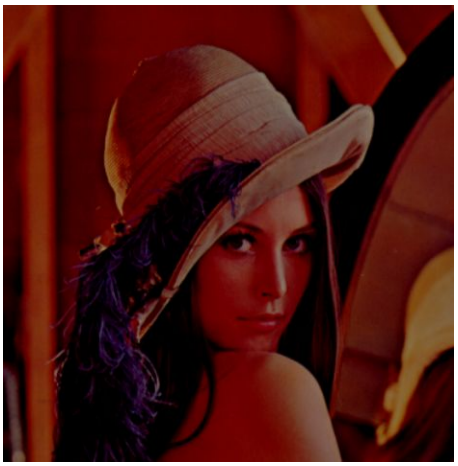
Brillantor

Es varia la lluminositat de la imatge. S'incrementa o es disminueix el valor de tots els píxels equitativament. Si s'augmenta el valor, la imatge serà més clara, mentre que si es disminueix, serà més fosca. [2]

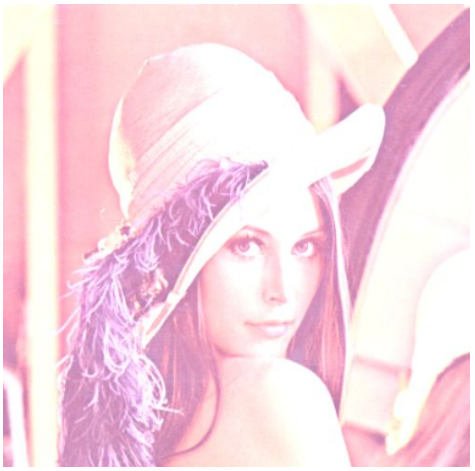
$$R' = R +- valor$$
$$G' = G +- valor$$
$$B' = B +- valor$$



Original.



Menys brillant.



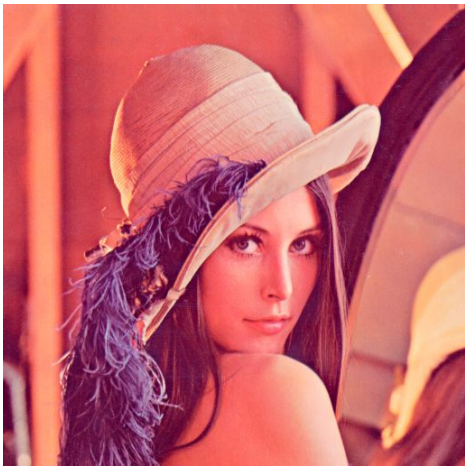
Més brillant.

Contrast

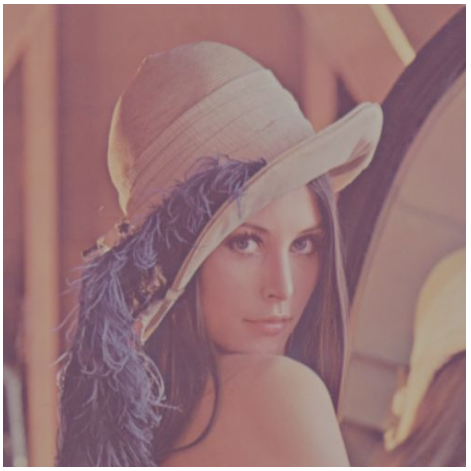
El contrast és la diferència de nivells d'intensitat entre un punt de la imatge i una zona pròxima a aquest. Quan hi ha poca diferència, el contrast és baix, mentre que quan hi ha molta diferència, el contrast és alt. Es varia multiplicant el valor de cada canal RGB dels pixels pel valor del canvi de contrast. [3]

$$R' = (((R / 255 - 0.5) * valor) + 0.5) * 255$$
$$G' = (((G / 255 - 0.5) * valor) + 0.5) * 255$$
$$B' = (((B / 255 - 0.5) * valor) + 0.5) * 255$$

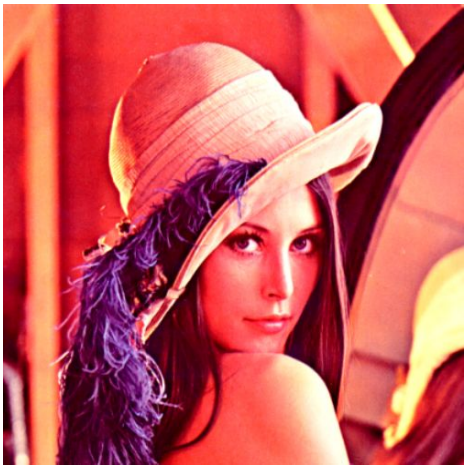
El resultat és una imatge en el qual hi ha una diferència més gran o petita entre les diferents zones de la imatge.



Original.



Menys contrastada.

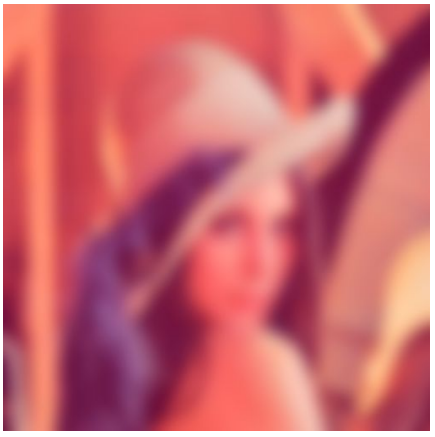


Més contrastada.

Desenfocar

Es genera un desenfoc per suavitzar les zones de la imatge on hi ha contorns, per donar la sensació que la imatge és més suau i simular falta de nitidesa. [4]

Original.

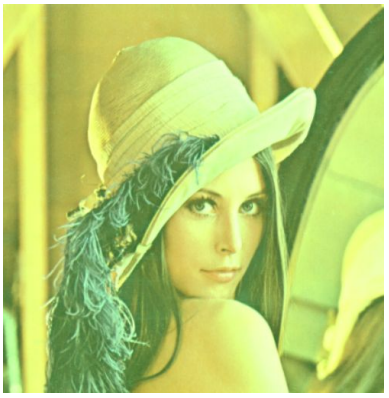


Desenfocada.

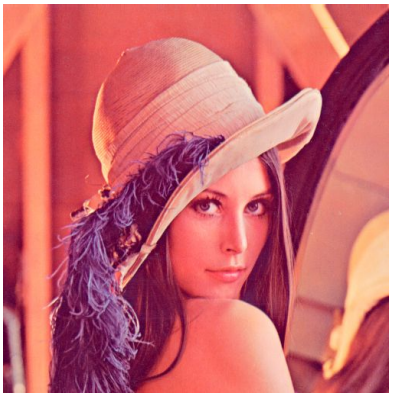
Equilibri de color

Es varia el nivell de cadascun dels diferents canals RGB de la imatge de manera independent.

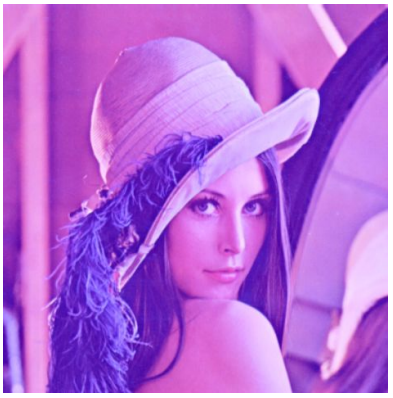
$$R' = R +- valorR$$
$$G' = G +- valorG$$
$$B' = B +- valorB$$



Component verda realçada.



Original.



Component blava realçada.

Invertir colors

S'inverteixen els valors dels canals RGB de la imatge dins del rang 0-255. [5]

$$R' = 255 - R$$
$$G' = 255 - G$$
$$B' = 255 - B$$

Original.

Invertida.

Zones il·luminades (Llums)

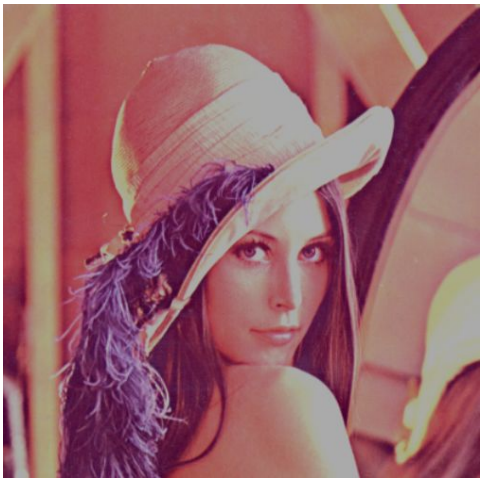
S'augmenta o es disminueix el valor de lluminositat únicament dels pixels que estan situats per damunt d'un determinat nivell, d'aquesta manera es pot controlar les parts de la imatge que són més clares. S'utilitza com a llindar separador de zones clares el valor 170. El que realitza aquest efecte realment és una compressió, de manera que quan el valor d'aplicació és el mínim, el rang de valors que es troba entre el 170 i el 255 es manté intacte. Quan el valor està a la meitat del seu rang d'actuació, els valors dels pixels d'aquest rang es veuen disminuïts en un factor de dos, és a dir, varien entre el 170 i el 212, i quan el valor és el màxim, tot el rang està limitat al valor 170.

Aquesta compressió s'obté aplicant la següent fórmula:

$$temp = (R - llindar\_llum) * valor$$
$$R' = llindar\_llum + temp * (255 - llindar\_llum) / (255 / 3)$$

Aquesta transformació pot ser aplicada tant a cadascuna de les tres components RGB com en el senyal de luminància.

Original.



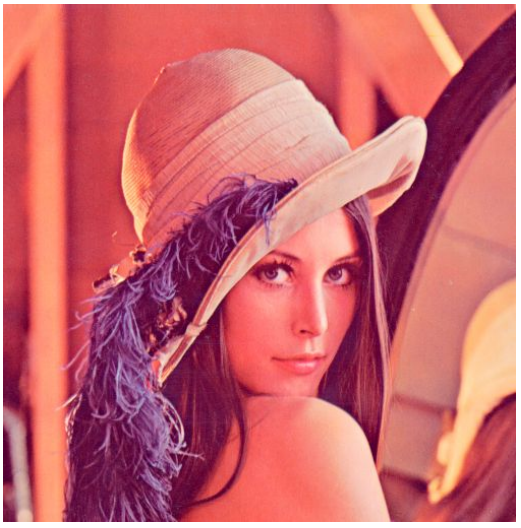
Il·luminacions baixades.

Nivell de blanc

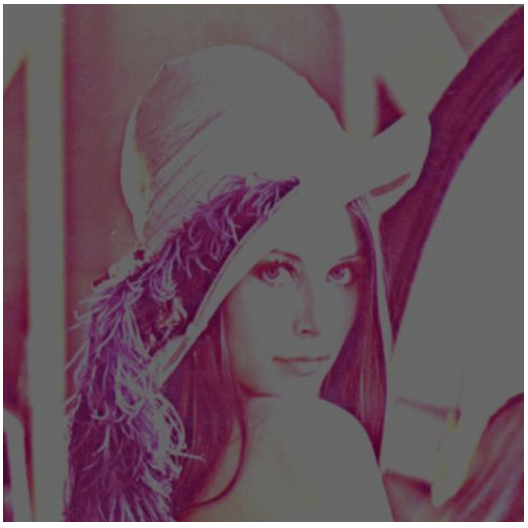
Es limita el valor més alt possible que pot tenir un pixel, fent que si aquest és superior al valor donat, se substitueix el valor del pixel pel valor definit per l'usuari.

Si R > valor, llavors R = valor

Aquesta transformació pot ser aplicada tant a cadascuna de les tres components RGB com en el senyal de luminància.



Original.



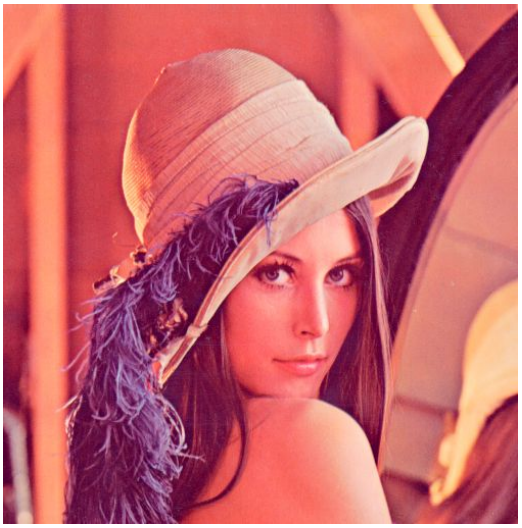
Nivell de blanc disminuït.

Nivell de negre

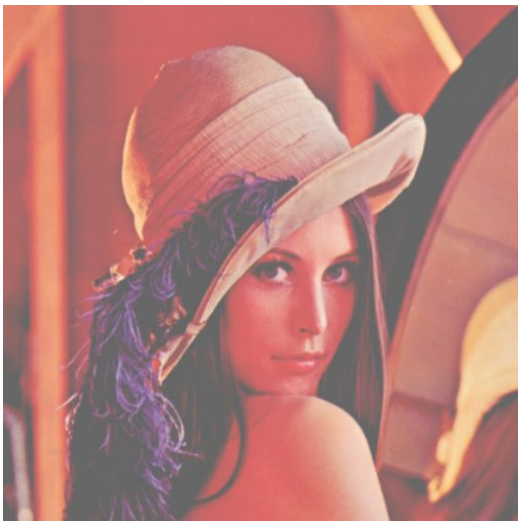
Es limita el valor més baix possible que pot tenir un pixel, fent que si aquest és inferior al valor donat, se substitueix el valor del pixel pel valor definit per l'usuari.

Si R < valor, llavors R = valor

De la mateixa manera que en l'efecte Nivell de blanc, aquesta transformació pot ser aplicada tant a cadascuna de les tres components RGB com en el senyal de luminància.



Original.



Nivell de negre augmentat.

Zones d'ombra (Ombres)

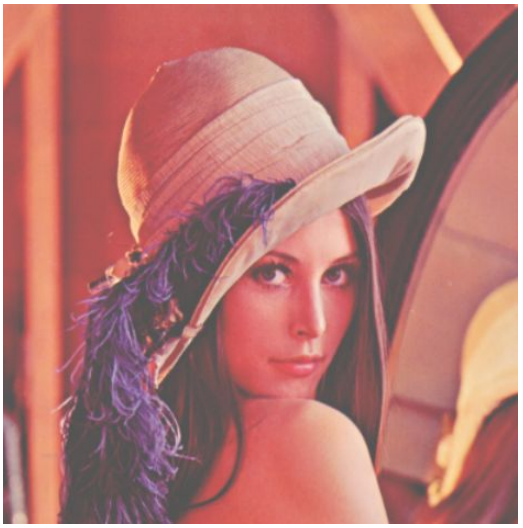
S'augmenta o es disminueix el valor de lluminositat únicament dels pixels que estan situats per sota d'un determinat nivell de valor, d'aquesta manera es pot controlar les parts de la imatge que són més fosques. S'utilitza com a llindar separador de zones fosques el valor 85. El que realitza aquest efecte realment és una compressió, de manera que quan el valor d'aplicació és el mínim, el rang de valors que es troba entre el 0 i el 85 es manté intacte. Quan el valor està a la meitat del seu rang d'actuació, els valors dels pixels d'aquest rang es veuen disminuïts en un factor de dos, és a dir, varien entre el 43 i 85, i quan el valor és el màxim, tot el rang està limitat al valor 85.

Aquesta compressió s'obté aplicant la següent fórmula:

$$R' = (R * valor + (llindar\_ombra - llindar\_ombra * valor))$$

Igual que en l'efecte Zones il·luminades (Llums), aquesta transformació pot ser aplicada tant a cadascuna de les tres components RGB com en el senyal de luminància.

Original.



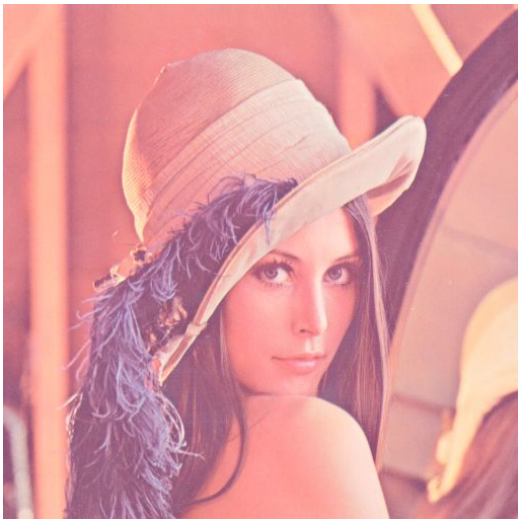
Ombres pujades.

Opacitat

Per a modificar l'opacitat d'una imatge es modifica la component alfa, que indica la transparència que té un pixel. Així doncs, el procediment a seguir és tan simple com assignar el valor que introdueix l'usuari al canal alfa de la imatge.

alfa = valor

Original.



Opacitat al 70%.

Retallar

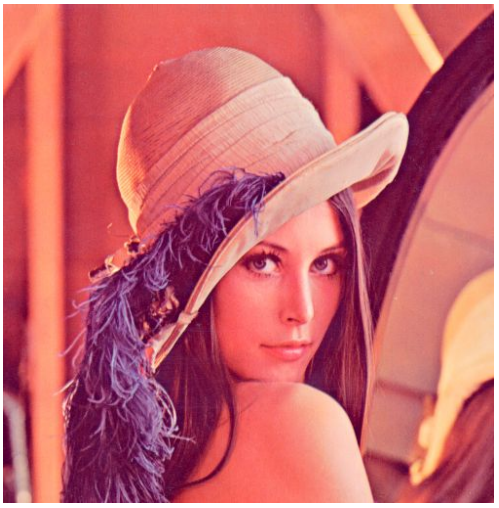
S'utilitza per reenquadrar i eliminar les parts de la imatge que no interessa mostrar. Es crea una nova imatge amb la mida desitjada i s'hi copien els valors que estan dins de l'àrea seleccionada en la nova imatge. Per seleccionar quina porció de la imatge es vol mantenir es mostra un requadre per indicar a l'usuari quina és la part de la imatge original que es conservarà. L'usuari pot modificar la mida i posició d'aquest requadre al seu gust.

Original.

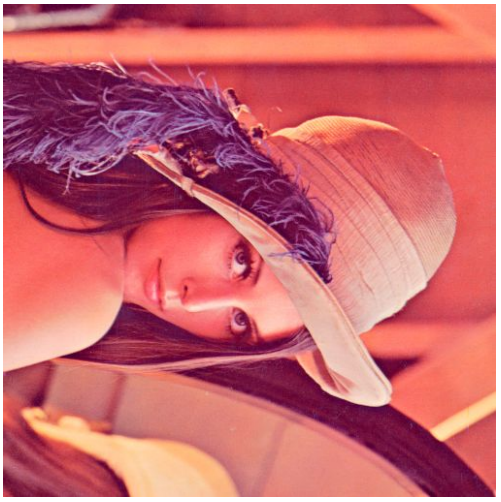
Retallada.

Rotació 90º

Gira la imatge 90° en sentit horari.



Original.



Rotada 90º.

Rotació d'angle

S'inclina la imatge un determinat nombre de graus definits per l'usuari. S'ha de tenir en compte que una imatge sempre ha de tenir forma rectangular i, quan s'aplica aquest efecte, per tal de mantenir aquesta relació, es corregeix aquesta deformació aplicant un re emplenament a les zones on aquesta no té cap valor de pixel assignat. Aquestes zones s'emplenen assignant-los el valor del color negre. [5]

Original.

Rotada.

Rotació horitzontal

S'aplica una rotació sobre l'eix Y (vertical) de la imatge, que comporta que la part situada a la dreta de la imatge es canviarà i es situarà a l'esquerra. L'eix X (horitzontal) es manté sense cap modificació. També s'anomena efecte mirall. [6]

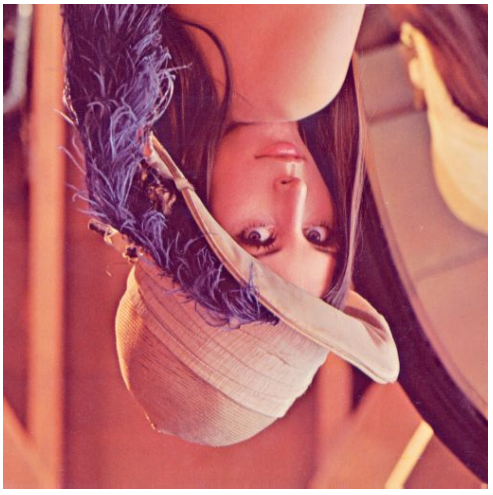
Original.

Rotada.

Rotació vertical

Igual que l'efecte anterior però en aquest cas la rotació s'aplica sobre l'eix X (horitzontal), intercanviant la part superior i inferior de la imatge.

Original.



Rotada.

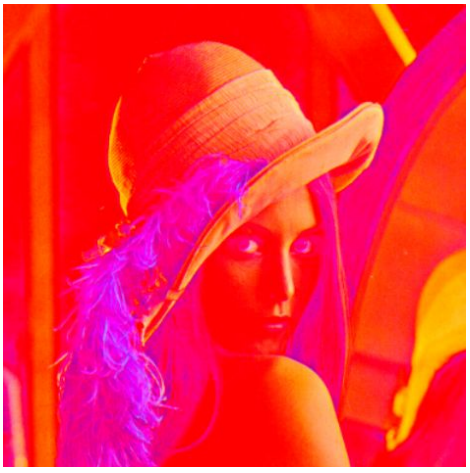
Saturació

És el nivell de puresa dels colors de la imatge, que està directament relacionat amb la quantitat de color blanc que té cada component de color. Quan es té una saturació elevada, els colors de la imatge són molt intensos, mentre que valors baixos provoquen que la imatge resulti amb uns colors apagats, fins a convertir-se a escala de grisos. S'aconsegueix poca saturació fent que els valors dels píxels dels diferents canals de color RGB siguin el més semblant entre ells, o el més diferent si es vol augmentar.

$$B\text{-}Y' = B\text{-}Y * valor$$
$$R\text{-}Y' = R\text{-}Y * valor$$

Original.

Menys saturació.



Més saturació.

2.2 Aplicacions d'Android

Android és un sistema operatiu dissenyat per a smartphones i tauletes. Ha estat desenvolupat majoritàriament per Google i és de codi obert. Això significa que el codi font del sistema es pot obtenir de manera gratuïta i qualsevol persona el pot modificar al seu gust i estudiar-ne el funcionament. El seu nucli base es Linux, altament versàtil i d'alta implantació en sistemes de servidors. Està bàsicament escrit en Java pel que fa a la interfície d'usuari i en C per la part del nucli.

Android va néixer l'any 2008 i des de llavors ha llençat al mercat diferents versions d'aquest software. Des de la primera versió, anomenada "Apple Pie" fins a l'actual, la versió 9.0, hi ha hagut un total de 16 versions, en les quals s'ha anat afegint diferents elements que permeten noves utilitats i millores, tant de seguretat com de tecnologies que últimament s'han posat de moda, com per exemple la *Comunicació de Camp Proper* (NFC, acrònim de l'anglès Near Field Communication), que fins fa relativament poc temps la gran majoria de gent desconeixia la seva existència i que és una tecnologia estandarditzada que permet la interconnexió entre dispositius propers.

En el món dels smartphones, Android és un dels grans en nombre d'usuaris, fins al punt en què la quota de mercat dels smartphones que utilitzaven el sistema operatiu de Google l'any 2017 era d'un 84% dels dispositius totals mundials. Els seus competidors directes són iOS, el famós sistema operatiu dels iPhone de la companyia americana Apple Inc. amb un 14% de la quota, i Windows Phone, de Microsoft, que es reparteix el 2% restant amb altres sistemes que en el seu temps van tenir molt de renom, com BlackBerry. A l'hora d'interpretar aquestes dades, s'ha de tenir compte que normalment els usuaris adquireixen un smartphone que incorpora un sistema operatiu Android o iOS, sense poder alternar entre aquests, i que el preu dels terminals que normalment incorporen Android acostuma a ser bastant reduït en comparació als telèfons iPhone.

Un aspecte important que s'ha de tenir molt en compte a l'hora de crear una aplicació és que les versions més noves d'Android acostumen a tenir una quota de mercat inferior a anteriors versions, i les versions més antigues pràcticament ja no s'utilitzen. Tot i que una versió mínima d'Android bastant baixa garanteix que l'aplicació pugui funcionar en una quantitat major de dispositius, no és molt convenient escollir-ne una d'aquest tipus, ja que alguns dels elements que es comentarà posteriorment, com per exemple el RecyclerView, no estan disponibles fins a certes versions, com l'API 22 en aquest cas.

De la mateixa manera, una versió molt nova, garanteix les últimes novetats pel que fa a elements disponibles, però es veu altament perjudicat el nombre de dispositius en què pot instal·lar-se l'aplicació. Aquest fet s'ha de tenir en compte a l'hora de decidir quina versió mínima d'Android s'ha d'elegir. El mateix Android Studio mostra una gràfica quan es crea una nova aplicació mostrant els percentatges estimats de quota de mercat que és compatible amb les diferents versions d'Android.

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99,6%
4.2 Jelly Bean	17	98,1%
4.3 Jelly Bean	18	95,9%
4.4 KitKat	19	95,3%
5.0 Lollipop	21	85,0%
5.1 Lollipop	22	80,2%
6.0 Marshmallow	23	62,6%
7.0 Nougat	24	37,1%
7.1 Nougat	25	14,2%
8.0 Oreo	26	6,0%
8.1 Oreo	27	1,1%

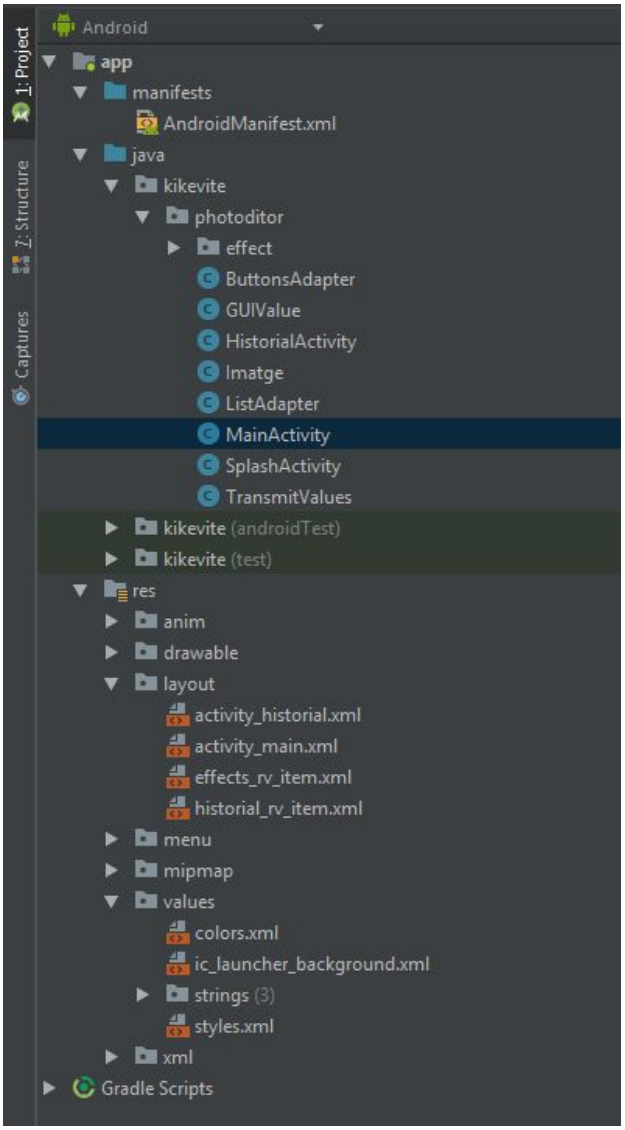
Taula de relació versió mínima/percentatge
de dispositius compatibles (a data 7/10/2018).

L'alta popularitat d'aquest sistema operatiu ha estat un punt més a favor de què el desenvolupament d'aquest projecte es realitzes en aquesta plataforma, enlloc de iOS o Windows Phone.

Una vegada s'ha decidit que el projecte es desenvoluparà per a plataformes Android s'ha d'indagar en com és l'estructura d'aquest i quins en són els components que la conformen, així com investigar sobre quins elements disposa de forma nativa. En els següents apartats es descriu aquests aspectes.

2.2.1 Components d'una aplicació d'Android

Un projecte d'aplicació d'Android consta habitualment de diferents tipus de fitxers, tot i que normalment segueixen un esquema bàsic, aquest pot acabar formant un projecte amb un nombre molt elevat d'arxius. En aquest projecte, el resultat final ha resultat en els apartats que es pot observar en la imatge que es mostra a continuació:



Exemple d'estructura de les components d'un projecte Android.

Tot seguit es llistarà quins són els tipus de components més importants, sense els quals una aplicació no podria funcionar, i quines funcions realitzen cadascun d'ells dins de l'aplicació.

Activitats

Una activitat [7] es pot descriure com cadascuna de les diferents pantalles amb les quals l'usuari interactua. Per exemple, en el cas de la coneguda aplicació WhatsApp, quan es prem la icona i s'obra l'aplicació, s'accedeix a l'activitat en què es mostren totes les converses. Quan es prem una de les converses, s'obra l'activitat que mostra una conversa. Cada activitat té un disseny visual diferent, anomenat layout. Això es veurà en posteriors seccions. Les activitats interactuen entre elles per tal de poder realitzar les funcions que l'aplicació requereix i permet marcar uns límits a l'hora d'assignar funcions, ja que realitzar massa accions en una sola pantalla pot comportar molts problemes de disseny, tant a nivell visual com a nivell d'implementació i funcionalitats.

El llenguatge que s'utilitza per a descriure el comportament de cadascuna de les diferents activitats es Java, àmpliament utilitzat tant en el desenvolupament de programes com en el de portals web, entre d'altres. S'ha de tenir en compte que no tots els arxius Java són activitats.

Serveis

Els serveis són operacions que es realitzen en segon pla mentre l'usuari completa altres tasques. La característica principal d'un servei és que no tenen una interfície amb la qual l'usuari interacciona de forma directa, tot i que habitualment l'execució d'un servei es produeix a causa d'accions que aquest vol realitzar. En l'exemple anterior de l'aplicació WhatsApp, un servei seria quan estem dins d'una conversa i rebem un missatge d'un altre usuari. Quan això passa, es llança un servei que actualitza els missatges que es mostren. Aquest exemple de servei no l'executa l'usuari de l'aplicació, i no té per què ser així sempre d'aquesta manera. De la mateixa manera que en les activitats, s'utilitza Java per a implementar-lo.

Recursos

Els recursos són una característica d'Android que permet externalitzar elements que s'utilitzen dins de l'aplicació, com per exemple, els textos, els layouts, o alguns valors. Els arxius de recursos tenen l'extensió xml, ja que estan escrits en llenguatge XML. Aquest és un meta-llenguatge descriptiu mitjançant etiquetes especialment pensat per a emmagatzemar dades. S'utilitza de manera extensa en el desenvolupament de pàgines web, per exemple, ja que proporciona una manera de guardar dades eficientment. L'exemple més clar el trobem en els textos que es mostren per pantalla. Un usuari pot tenir el smartphone configurat en català i un altre en anglès, per tant, l'aplicació ha de mostrar els textos en els respectius idiomes de cada usuari. Si aquests textos s'incrusten dins del codi, no hi ha manera possible de traduir-los i mostra un text comprensible per a tots els usuaris. És aquí quan entren en joc els recursos.
Organitzats en diferents categories, els recursos permeten fer referències a parts del codi que sovint canvien, sigui perquè s'han de traduir o perquè han de variar segons altres característiques del smartphone, com pot ser la mida de la pantalla.

Trobem els recursos classificats de la següent manera:

- animator: arxius d'animacions de propietats.
- anim: arxius que descriuen animacions d'interpolació de moviment.
- color: arxius que defineixen estats de diferents colors.
- drawable: bàsicament, arxius d'imatges o que intenten realitzar una funció similar a la d'una imatge, com per exemple descriure el disseny d'un element.
- mipmap: elements dels quals es compon el llançador de l'aplicació (icona) per a tots els tipus de resolucions de pantalla.
- layout: arxius que descriuen la composició visual de cada activitat o de Views específics. Són els que es fa referència en l'apartat anterior, on s'explica que és una activitat.
- menu: definició dels layouts específics per a menús.
- raw: arxius que s'emmagatzemen sense processar.
- values: són el tipus de recurs que serveixen per evitar "hardcodejar" els valors de certes variables, com per exemple Strings traduïdes, valors de colors i tipus d'estils, entre altres.
- xml: altres tipus d'arxius que no estan descrits anteriorment.

Android Manifest

El manifest [8] és un dels arxius més importants de l'aplicació. En ell es configura la majoria de paràmetres d'aplicació i s'hi descriu el contingut. Alguns d'aquests paràmetres poden ser identificar l'app de manera única, enumerar les diferents activitats que formen l'app, els permisos d'usuari que es requereixen, les biblioteques que han d'estar vinculades o la versió mínima d'Android requerida per a poder executar l'app.

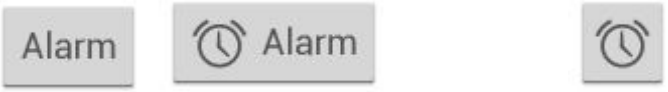
2.2.2 Elements d'android

Per a poder efectuar les accions que es vol desenvolupar en aquesta aplicació, s'han d'utilitzar alguns dels elements ja creats i que proporciona de manera nativa Android. Per a poder utilitzar-los, s'ha realitzat un estudi sobre les necessitats de l'aplicació i s'ha investigat sobre quins elements dels quals proporciona Android compleixen els requisits per a poder realitzar les diferents tasques que es té pensat implementar. El resultat d'aquesta investigació es plasma a continuació.

Button

Els buttons [9] són uns dels elements més simples i utilitzats a l'hora de crear una interacció entre l'usuari i l'aplicació. Són un element que permet realitzar accions cada vegada que l'usuari interacciona amb ell fent-li clic. Segons el tipus de contingut visual que contingui el botó, podem distingir-ne dos tipus diferents:

- Button: botons formats per text únicament o una imatge i text de forma conjunta.
- ImageButon: botons formats exclusivament per una imatge.



Exemple de Buttons i ImageButton.

Per crear un Button en una activitat s'afegeix el següent codi al fitxer XML del layout de l'activitat en la qual volem que aquest aparegui:

```
<Button
   android:id="@+id/idButton"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content" />
```

Una vegada s'ha creat el botó, se li ha d'assignar el codi que es vol executar en fer-li clic. Podem fer-ho també de dues maneres diferents, però bastant semblants:

La primera és afegint en la definició del botó en el fitxer XML anteriorment citat l'atribut 'onClick', en la qual se li assigna a aquest botó una funció, que serà la que es cridarà cada vegada que el botó es premi:

android:onClick="sendMessage"

En conseqüència, en l'arxiu Java corresponent a aquesta activitat, ha d'haver-hi la funció corresponent:

```
public void sendMessage(View view) {
    // Do something in response to button click
}
```

En el segon mètode s'utilitza un 'onClickListener'. Un listener és un element que està pendent de tot el que li passa a l'element que té associat. Hi ha molts tipus diferents de listeners, tals com els que detecten quan es prem una tecla, quan es canvia l'orientació del dispositiu, etc. En el cas que ens interessa, utilitzarem un listener de tipus 'onClick', i l'assignarem al botó realitzant les accions següents:

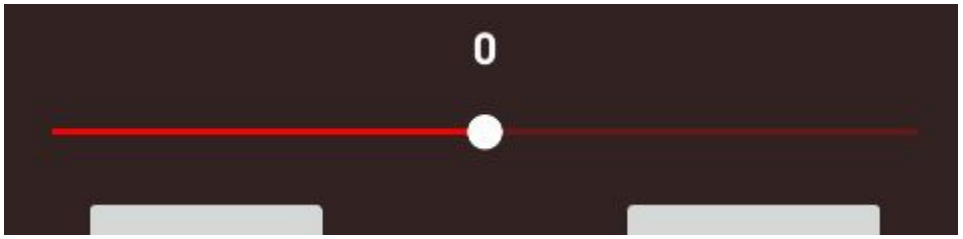
```
Button button = (Button) findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
    }
});
```

Primer es crea una referència al botó, ja que per assignar un listener es necessita un objecte que sigui de la classe View. S'ha de tenir en compte que el mètode 'setOnClickListener()' pertany a la classe View, no a la Button, i que la classe Button és una subclasse de View.
Això ho fem utilitzant el mètode 'findViewById()', que retorna la referència a l'element View que es passa com a paràmetre d'entrada a la funció. Aquest element ha d'estar creat dins l'arxiu XML. Seguidament s'invoca el mètode 'setOnClickListener()' sobre aquest. Com a paràmetre d'entrada d'aquest mètode se li passarà el listener desitjat. En aquest cas serà del tipus 'View.OnClickListener' i se sobreescriurà el mètode 'onClick()', en el qual implementarem el codi desitjat.

La principal diferència entre aquests dos mètodes és que en el primer cas no és necessària la creació de l'objecte de la classe Button ni obtenir-ne la referència. Això és necessari per exemple quan es crea el botó en un instant diferent de l'inici de l'aplicació.

SeekBar

Una SeekBar [10] és una barra que té la finalitat d'assignar un valor a una variable. Consisteix en una barra horitzontal i un punt anomenat 'thumb' que l'usuari pot moure per tal de variar-ne el valor. Aquest pot trobar-se en el seu valor mínim quan el thumb de la SeekBar està situat al límit esquerre de la barra horitzontal, i en el seu valor màxim quan el thumb es troba en la posició oposada, és a dir, al límit dret de la barra horitzontal. S'utilitza per assignar un valor variable mitjançant un element visual. Normalment també s'acostuma a mostrar quin és aquest valor mitjançant un text, per tal que l'usuari tingui el coneixement de si està incrementant, disminuint o simplement saber-ne el valor.



Exemple de SeekBar amb un TextView per mostrar-ne el valor.

Per tal d'utilitzar una SeekBar, primer s'ha de crear en l'arxiu XML mitjançant el següent codi:

```
<SeekBar
        android:id="@+id/seekBar"
        android:layout_width="300dp"
        android:layout_height="wrap_content" />
```

Amb els atributs 'min' i 'max' s'estableix quins són els valor mínim i màxim que pot assolir la SeekBar. Per defecte, aquests són 0 i 100 respectivament. Aquests atributs es poden ometre en el moment de la creació de la SeekBar, ja que posteriorment es poden assignar.
Seguidament es procedeix obtenint la referència a l'element de la mateixa manera que en el cas del botó, utilitzant el mètode 'findViewById()' i es continua assignant-li un listener del tipus 'onSeekBerChangeListener':

```
bar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
        }

                @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
        }
});
```

Aquest listener és una interface que ens obliga a implementar 3 mètodes:

- onProgressChanged(): accions que es realitzen quan es varia el valor assignat a la SeekBar a través del mètode corresponent o quan es mou el thumb.
- onStartTrackingTouch(); accions que es realitzen quan es prem el thumb, una sola vegada a l'inici.
- onStopTrackingTouch(): accions que es realitzen quan es deixa de prémer el thumb, una sola vegada al final.

En el cas que no es vulgui realitzar cap acció en algun dels tres mètodes, simplement es deixa el cos del mètode buit, però mai es pot deixar sense sobreescriure, ja que es tracta d'una interface i forçosament s'ha d'implementar tots els mètodes que aquesta implementa.

En els dos últims mètodes, 'onStartTrackingTouch()' i 'onStopTrackingTouch()', únicament es rep com a paràmetre l'objecte SeekBar, a diferència del mètode 'onProgressChanged()', que rep també els elements 'int i' i 'boolean b', que indiquen respectivament el valor de la SeekBar i si el canvi de valor ha estat conseqüència de l'usuari o no. Això és necessari, ja que es pot invocar el mètode 'setProgress()' per a assignar-li un valor concret.

Com s'ha comentat anteriorment, els valors mínim i màxim es poden modificar, mitjançant els mètodes 'setMin()' i 'setMax()'.
També és possible obtenir aquests dos valors amb els corresponents getters, 'getMin()' i 'getMax()', així com obtenir el valor de la SeekBar en qualsevol moment mitjançant el mètode 'getProgress()' tal com s'ha mencionat.

ImageView

La classe ImageView [11] està destinada a la visualització d'imatges. Normalment aquestes imatges estan emmagatzemades dins d'objectes de les classes Bitmap o Drawable, per exemple, tot i que no sempre ha de ser així. En aquesta ocasió ens centrarem en el cas dels Bitmap [12], ja que és l'opció per la qual s'ha optat per a emmagatzemar les imatges en aquest projecte, tal com es veurà més endavant.

Per crear un ImageView implementarem el següent codi a l'arxiu XML:

```
<ImageView
        android:id="@+id/idImageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher"
        />
```

L'atribut 'src' indica quina imatge es vol mostrar. En aquesta ocasió no l'utilitzarem, ja que hi carregarem un Bitmap a través de l'arxiu Java de la corresponent activitat.

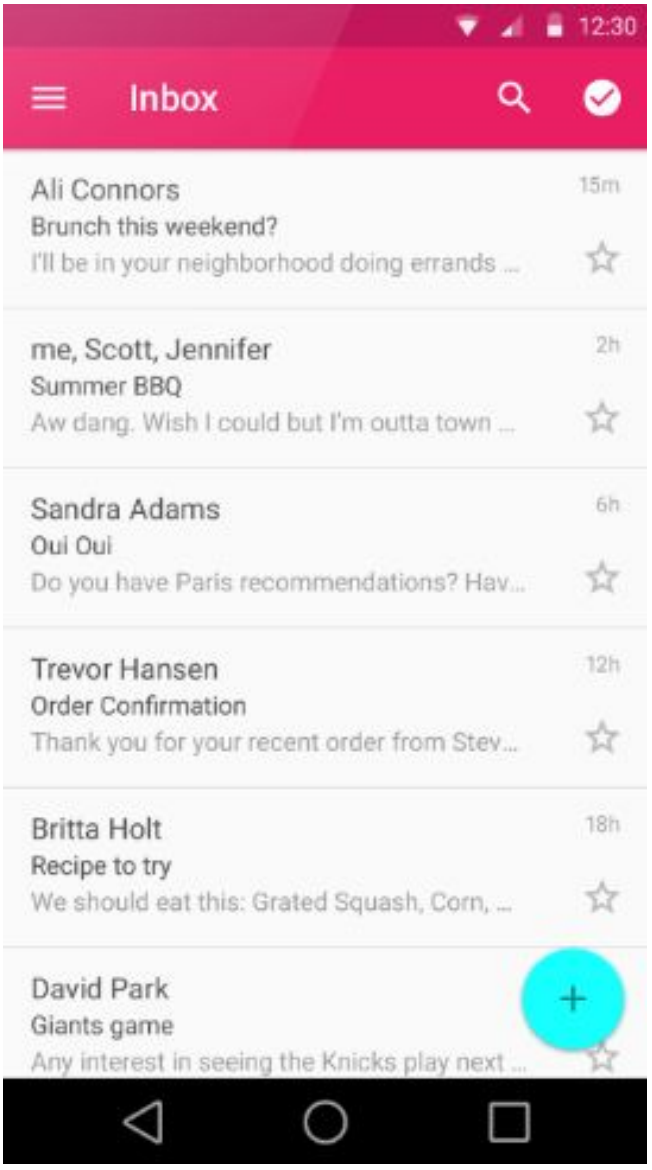
L'atribut 'background' permet assignar un color al ImageView. Posteriorment a què es carregui una imatge al ImageView, si els píxels corresponents no estan ocupats per la imatge carregada, aquests es mostraran d'aquest color.

També és important l'atribut 'scaleType', ja que estableix com s'adapta la imatge que es vol mostrar dins dels límits del ImageView. Alguns dels valors disponibles són: CENTER (centra la imatge al mig del ImageView sense redimensionar), FIT_CENTER (centra la imatge dins del ImageView i la redimensiona fins a adaptar-se a les mides d'aquest) o MATRIX (escala la imatge segons l'objecte de la classe Matrix que se li assigna).

També s'utilitza el mètode 'setOnTouchListener()' en el qual se li assigna un listener al ImageView. En el nostre cas li assignem un listener del tipus 'onTouch', ja que, com es veurà en el disseny de l'app, es vol que es realitzin certes accions quan l'usuari toqui el ImageView. Per fer-ho se sobreescriu el mètode 'onTouch()' de l'activitat en la qual es troba el ImageView. Posteriorment es fa que l'activitat implementi l'interface 'View.OnTouchListener' [13]i finalment s'assigna al ImageView el listener amb el mètode anteriorment nomenat 'setOnTouchListener(this)'.

RecyclerView

Un RecyclerView [14] és un objecte contenidor d'altres objectes. Serveix per mostrar llistes d'elements amb necessitats generalment complexes. Amb complexos ens referim a elements que estan formats per dades que canvien constantment de format o que estan formats per altres Views. La necessitat d'utilitzar un RecyclerView apareix a causa de les limitacions que tenen els ListView, més fàcils i intuïtius d'utilitzar. La principal diferència recau sobre l'àmplia personalització que es pot aplicar als RecyclerView i que limita als ListView.



Exemple de RecyclerView.

Per afegir un RecyclerView s'utilitzarà el següent codi a l'arxiu XML, però abans d'això, s'ha d'afegir una llibreria de suport a la part de les dependències de l'arxiu build del Gradle. Aquesta és la 'v7 Support Libraries', i d'aquesta manera podrem accedir al codi que implementa el RecyclerView correctament:

```
dependencies {
        implementation 'com.android.support:recyclerview-v7:28.0.0'
}
```

```
<android.support.v7.widget.RecyclerView
        android:id="@+id/idRecyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"/>
```

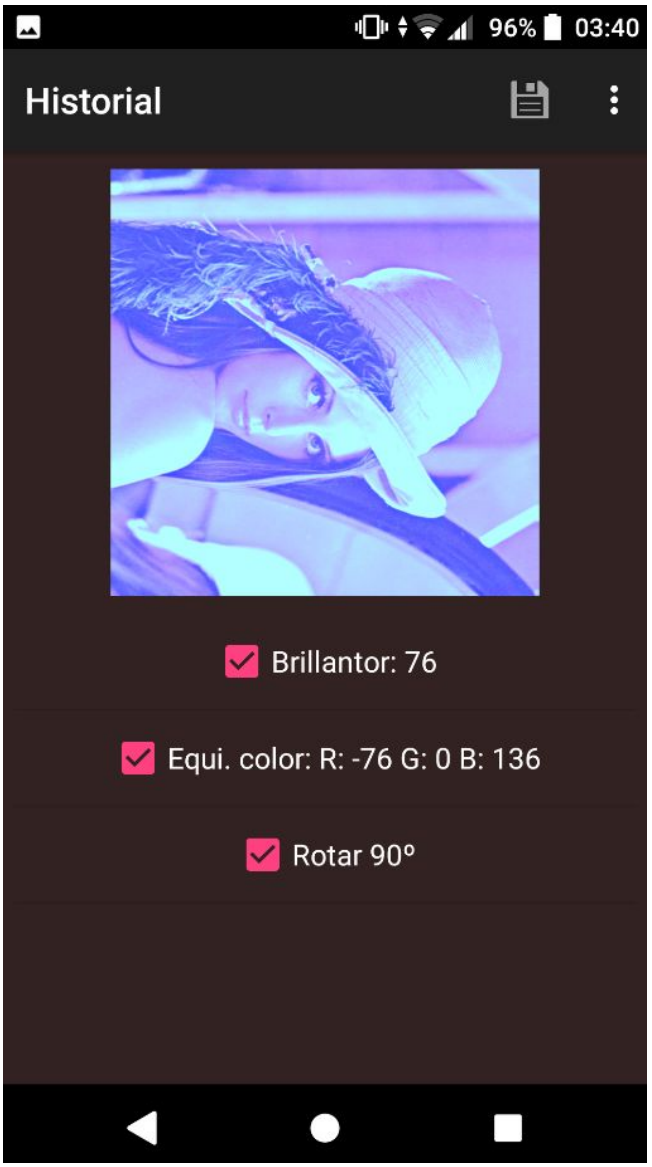
En ser el RecyclerView un objecte tan personalitzable, s'ha de crear un 'Adapter' a mida, que serà l'encarregat d'assignar a cada casella del RecyclerView les dades que li pertoca.

RecyclerView.Adapter

Es crearà una nova classe que extengui la classe RecyclerView.Adapter<> [15]. Aquesta classe ha de tenir les següents parts:

- Com a atributs, les dades que es volen mostrar i un listener, juntament amb una interface que s'assignarà a l'activitat com a listener.
- Un constructor en el qual es passaran com a paràmetres les dades que es mostraran i el listener que utilitzarà.
- Classe interna ViewHolder que extengui la classe RecyclerView.ViewHolder.
- Sobreescriure el mètode 'onCreateViewHolder()'.
- Sobreescriure el mètode 'onBindViewHolder()'.
- Sobreescriure el mètode 'getItemCount()'.

Per a exemplificar la utilització d'aquests mètodes, s'utilitzarà com a exemple el ReciclerView de l'activitat HistorialActivity.



RecyclerView de HistorialActivity.

Aquest RecyclerView mostra un llistat d'efectes i si aquests es troben aplicats sobre una imatge o no. Per plasmar-ho, s'utilitza un CheckBox en el qual el text indica el nom de l'efecte juntament amb el valor d'aplicació d'aquest i el requadre si l'efecte es troba actualment aplicat.

En aquest cas les dades a mostrar es troben en un array del tipus boolean (per marcar l'efecte com a aplicat o no) i en un ArrayList<Effect> anomenat 'historial', d'on obtindrem el nom de l'efecte i el valor d'aquest:

private ArrayList<Effect> historial;
private boolean[] effectStatus;

Seguidament es crea una interface que actuarà com a listener del RecyclerView. Aquesta conté un mètode que passa com a paràmetre la posició de la casella que s'ha clicat en el RecyclerView. Aquest mètode serà el que s'implementarà a l'activitat corresponent i el que es cridarà cada vegada que l'usuari faci clic el RecyclerView:

```
public interface OnCheckBoxBarClickedListener {
        void onCheckBoxBarClicked(int pos);
}
```

Es crea el corresponent atribut d'aquesta interface:

```
private OnCheckBoxBarClickedListener listener;
```

En el constructor s'assignaran les dades que es mostraran en el RecyclerView, i també el listener que s'associa amb aquest:

```
ListAdapter(ArrayList<Effect> historial, boolean[] effectStatus,
OnCheckBoxBarClickedListener listener) {
   this.historial = historial;
   this.effectStatus = effectStatus;
   this.listener = listener;
}
```

En la classe interna 'ViewHolder' es crearan les referències als Views que conté cada casella del RecyclerView. En el nostre cas únicament es requereix un CheckBox, per tant es crea l'objecte CheckBox i n'obtenim la referència amb 'findViewById()' com ja s'ha fet anteriorment:

```
static class ViewHolder extends RecyclerView.ViewHolder {
   private CheckBox checked;
   ViewHolder(View v) {
      super(v);
      checked = v.findViewById(R.id.checkBox);
   }
}
```

El mètode 'onCreateViewHolder()' és l'encarregat de crear el contingut (els Views) de les caselles. Per fer-ho, s'utilitza un LayoutInflater en el qual, com a paràmetre, es passa la referència de l'arxiu XML que conté el disseny de cada casella que es crearà:

```
@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View v = LayoutInflater.from(parent.getContext()).inflate(R.layout.historial_rv_item, parent,
false);
    // Configuració del View 'v'
    return new ViewHolder(v);
}
```

Aquest és el codi de l'arxiu XML historial_rv_item que conté cada element CheckBox:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="10dp">

    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="16sp" />

</RelativeLayout>
```

Layout de cada casella del RecyclerView.

Un RecyclerView, com el seu nom indica, està pensat per a tenir un funcionament que recicla els Views que utilitza. Això s'implementa de la següent manera. Es creen el nombre de caselles necessàries per a emplenar el RecyclerView que es mostra per pantalla i algunes més per a tenir-les preparades en cas que l'usuari les hagi de visualitzar. Quan l'usuari fa scroll per veure les següents caselles que estan ocultes, es mostren les caselles que es tenien en reserva i es posen en reserva les que deixen de mostrar-se. Per exemple, en una llista de 100 elements, el RecyclerView no crea 100 caselles, sinó que únicament crea el nombre de caselles que es mostren en la pantalla, com per exemple 5, i unes quantes més, que podrien ser la següent a mostrar i l'anterior. D'aquesta manera s'estalvia molt de processament. Quan una casella es mostra es podria dir que les seves dades estan fixades, però quan entra en reserva, les dades poden ser substituïdes per les dades d'una casella candidata a ser mostrada. És per això que el RecyclerView recicla les caselles que mostra.

El mètode 'onBindViewHolder()' s'encarrega de reciclar els elements de cada casella. Agafa les dades corresponents i les fica dins de cada View de cada casella del RecyclerView. Això ho fa tant en les caselles que es mostren des de la creació del RecyclerView com en les que estan en reserva. En el nostre exemple, els requadres dels CheckBox s'emplenen amb les dades de l'array de booleans, mentre que el text s'emplena amb les dades de l'ArrayList:

```
@Override
public void onBindViewHolder(ViewHolder holder, final int pos) {
    holder.checked.setText(historial.get(pos).toString());
    holder.checked.setChecked(effectStatus[pos]);
    holder.checked.setTextColor(0xffffffff);
    holder.checked.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (listener != null) {
                listener.onCheckBoxBarClicked(pos);
            }
        }
    });
}
```

Per acabar amb la implementació de l'adaptador propi, el mètode 'getItemCount()' retorna el nombre total de caselles que hi ha en el RecyclerView. En aquest cas es retorna la mida de l'historial:

```
@Override
public int getItemCount() {
        return historial.size();
}
```

Una vegada finalitzada la creació de la classe adaptador per al nostre RecyclerView, es procedeix amb la configuració d'aquest en el fitxer Java de l'activitat. En aquesta ocasió s'han utilitzat els següents mètodes per a configurar-lo:

Primer s'obté la referència amb el mètode 'findViewById()'. Seguidament, se li assigna una mida fixa amb el mètode 'setHasFixedSize()'. Això significa que la modificació de les dades que es mostraran en el RecyclerView no provocaran que l'adaptador d'aquest hagi de canviar les mides que té el RecyclerView en la pantalla del telèfon de l'usuari. En aquesta ocasió això es compleix, per tant se li passa com a paràmetre a aquest mètode un 'true'.

Es realitza una crida al mètode 'setLayoutManager()' amb la finalitat d'assignar al RecyclerView si ha de mostrar una llista vertical o horitzontal. En l'exemple mostrat, s'utilitza un layout vertical, per tant, es passa com a paràmetre d'entrada a aquesta funció un objecte de la classe RecyclerView.LayoutManager de la subclasse LinearLayoutManager, que configurarem perquè tingui un aspecte vertical:

```
RecyclerView.LayoutManager myLayoutManager = new
LinearLayoutManager(getApplicationContext(), LinearLayoutManager.VERTICAL, false);
myRecyclerView.setLayoutManager(myLayoutManager);
```

Posteriorment s'afegeix un element separador entre les diferents caselles que es mostren. Utilitzarem el mètode 'addItemDecoration()' passant com a paràmetre un nou objecte de la classe DividerItemDecoration [16] configurat en posició vertical:

```
myRecyclerView.addItemDecoration(new DividerItemDecoration(this,
LinearLayoutManager.VERTICAL));
```

Finalment, només ens falta crear l'adaptador de la classe que s'ha creat anteriorment i utilitzar el constructor. Assignarem aquest adaptador al RecyclerView amb el mètode 'setAdapter()':

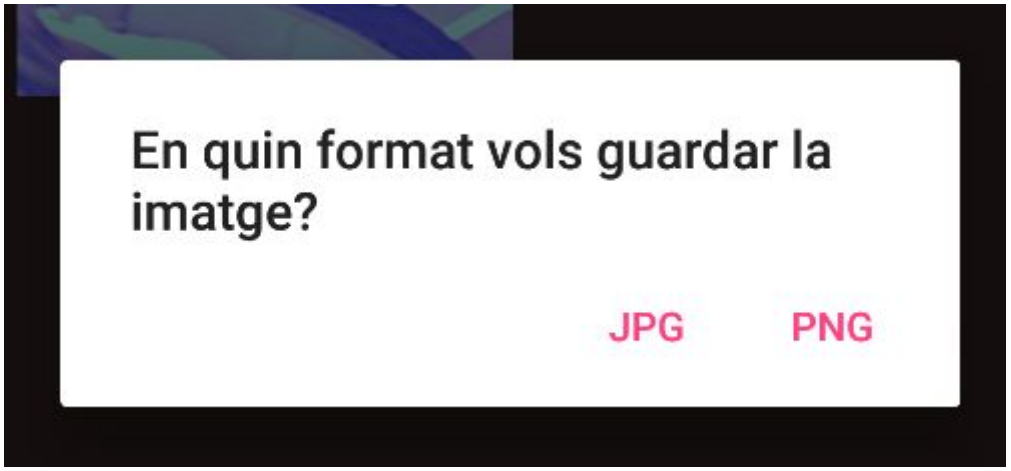
```
myAdapter = new ListAdapter(historial, effectStatus, this);
myRecyclerView.setAdapter(myAdapter);
```

S'ha de tenir en compte que quan es modifiquen les dades que es mostren en el RecyclerView, s'ha d'avisar a l'adaptador sobre aquest fet, cridant al mètode de l'adaptador 'notifyDataSetChanged()'. Això actualitza les dades que es mostren per pantalla en el RecyclerView.

Un altre dels mètodes que s'ha utilitzat en el RecyclerView és el mètode 'scrollToPosition()'. Passant com a paràmetre un nombre enter, configura el RecyclerView perquè es mostri la posició desitjada. Per exemple, si es vol mostrar la primera posició, aquest valor serà el 0.

AlertDialog

Els AlertDialog [17] són uns elements que tenen la funció d'informar a l'usuari sobre algun fet i fer que prengui alguna decisió. S'utilitzen per mostrar informació detallada a l'usuari o perquè aquest introdueixi dades d'una manera destacada, ja que el missatge es mostra en una petita finestra semblant als pop-ups dels navegadors d'Internet.



Exemple d'AlertDialog.

La classe AlertDialog és la més personalitzable, ja que es pot definir quins elements es vol mostrar en el quadre de diàleg. Altrament, hi ha dos altres tipus de quadres de diàleg que ja estan pensats per dur a terme funcions més específiques. Aquests són DatePickerDialog i TimePickerDialog, que com el seu nom indica, estan pensats perquè l'usuari introdueixi una data o una hora.
En el nostre cas utilitzarem la classe AlertDialog, ja que no es necessita en cap moment que l'usuari esculli alguna de les dues opcions anteriors.

Un quadre de diàleg estàndard està format per les següents parts:

- Títol: text que normalment resumeix les intencions que es volen dur a terme.
- Cos: àrea on es mostra l'acció que l'usuari ha de realitzar. Pot ser, per exemple, un text amb una pregunta o una llista en la qual l'usuari ha d'escollir una opció.
- Botons: Per acceptar o cancel·lar les accions a realitzar. N'hi ha de tres tipus diferents: Positiu, normalment per acceptar els canvis que proposa el cos del AlertDialog, negatiu, per cancel·lar-los, i neutral, que deixa la proposta del cos en standby, és a dir, ni cancel·la ni accepta la proposició.

A diferència dels elements anteriors, AlertDialog no és un View de la mateixa manera que els Buttons o els ImageViews, i per tant, no s'ha de declarar en l'arxiu XML de l'activitat. Únicament es declara en l'arxiu Java de l'activitat, ja que es mostra en un moment determinat i no està constantment visible en la pantalla de l'usuari.

Per fer aparèixer un quadre de diàleg, s'ha de seguir els següents passos:

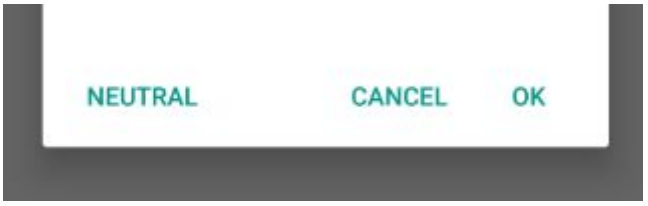
En lloc de crear un objecte de la classe AlertDialog, se'n crea un de la classe AlertDialog.Builder [18], que fent el símil amb la classe explicada anteriorment RecyclerView, seria com l'adaptador que s'utilitza per a tractar les dades que es mostra i crear la vista. Per tant es crea un objecte d'aquesta classe passant com a paràmetre l'activitat en la qual s'ha de mostrar aquest:

```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
```

Seguidament es procedeix configurant un missatge al cos del quadre, si és el que es vol mostrar, i afegint un títol a aquest. Tot i que aquests camps no són obligatoris, es recomana mostrar-ne almenys un. Això ho fem amb els mètodes 'setMessage()' i 'setTitle()' respectivament, passant com a paràmetre els Strings que es volen mostrar:

```
builder.setMessage(R.string.dialog_message);
builder.setTitle(R.string.dialog_title);
```

Després, s'assignen els botons que es volen mostrar. Els tres botons es mostren en la part inferior del quadre de diàleg:



Exemple de botons de l'AlertDialog.

Per afegir-los s'utilitza els mètodes 'setPositiveButton()', 'setNegativeButton()' i 'setNeutralButton()'. Tots tres segueixen el mateix esquema de funcionament: com a primer paràmetre reben l'String que es mostra en el text del botó i com a segon un listener que executa el codi contingut dins seu quan es fa clic:

```
builder.setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
          // Code
   }
});
```

Una vegada realitzada la configuració dels elements que conformen el AlertDialog es crida el mètode 'create()' sobre el builder, seguit del mètode 'show()' perquè el quadre de diàleg es mostri:

```
builder.create().show();
```

Quan es mostra el quadre, només s'ha d'esperar la interacció de l'usuari per procedir executant les tasques segons l'elecció de l'usuari.

Toast

Un Toast [19] és un quadre informatiu molt bàsic. Únicament mostra un text durant uns quants segons, en una àrea de petit tamany en la pantalla. La seva configuració és tan senzilla que en una sola línia de codi es pot executar la visualització d'un Toast bàsic.

Exemple de Toast.

De la mateixa manera que els AlertDialog, per a crear un Toast no és necessari implementar codi en l'arxiu XML de l'activitat en la qual es vol mostrar.

S'utilitza el mètode estàtic de la classe Toast 'makeText()' que rep com a paràmetres el context en el quan es vol mostrar el missatge, el String que es vol mostrar, i la duració d'aquest missatge en pantalla, que està predefinit en dues opcions com a atributs de classe: Toast.LENGTH_SHORT i Toast.LENGTH_LONG. Finalment, per a mostrar-lo, s'invoca el mètode 'show()' sobre aquest:

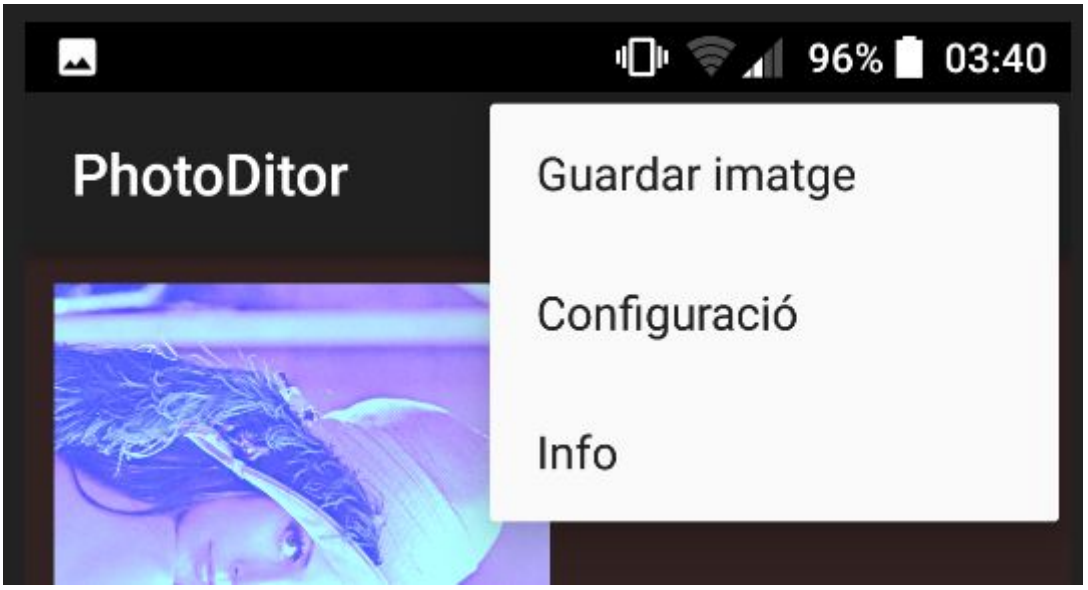
```
Toast.makeText(this, R.string.string, Toast.LENGTH_SHORT).show();
```

Tot i ser un element tan senzill, té algunes funcions de personalització, com per exemple la posició en la pantalla. Per canviar-la, s'ha d'utilitzar la classe Gravity [20] i el mètode 'setGravity()' de Toast. Primer es crea un objecte d'aquesta classe, en el quan li assignarem el text i la duració. Posteriorment li assignarem la posició i finalment el mostrarem amb 'show()':

```
Toast toast;
toast = Toast.makeText(activity, activity.getString(R.string.not_saved),
Toast.LENGTH_LONG);
toast.setGravity(Gravity.CENTER_VERTICAL | Gravity.CENTER_HORIZONTAL, 0, 500);
toast.show();
```

Menu

Un Menu [21] és un llistat desplegable d'accions que normalment està situat a la barra de l'aplicació. Si troben funcions que l'usuari ha d'executar amb relativa freqüència i per tant han de situar-se en una posició fàcilment accessible.



Exemple de menu.

Per a la creació del menu s'ha de crear un arxiu xml en la carpeta menu dels recursos. Aquest arxiu descriu cadascun dels elements que apareixeran dins del menú, anomenats items. Únicament se'n descriu el contingut, les accions que aquests realitzen es configuren a l'activitat.
Cada item tindrà una estructura semblant a la següent:

```
<item
    android:id="@+id/action_load"
    android:icon="@android:drawable/ic_menu_camera"
    android:orderInCategory="1"
    android:title="@string/load_image"
    app:showAsAction="ifRoom" />
```

Amb la propietat 'icon' s'assigna la icona a l'item del menu, amb 'orderInCategory' la posició en què es mostrarà dins del llistat, si aquest es vol mostrar en algun ordre determinat, i amb 'showAsAction' es decideix si l'item s'ha de mostrar sempre, només quan hi ha espai suficient en la pantalla o si sempre ha d'anar amagat dins del desplegable d'opcions que no es poden mostrar, sigui per tamany de pantalla o obligació.

Per fer aparèixer el menú, en l'activitat s'ha de sobreescriure el mètode 'onCreateOptionsMenu()'. Aquest mètode és l'inflador del menú i es passarà com a paràmetre el recurs xml del menú anteriorment creat.

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_menu_layout, menu);
    return super.onCreateOptionsMenu(menu);
}
```

Per a assignar cadascuna de les funcions a executar en fer clic a cada ítem se sobreescriu el mètode 'onOptionsItemSelected(MenuItem item)'. En la propietat id del item, que obtenim a través del mètode 'getItemId()', hi ha l'id de l'ítem que s'ha creat, dada que s'aprofita per identificar quines accions s'ha de fer.

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_1:
            // Action 1
            return true;
        case R.id.action_2:
            // Action 2
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

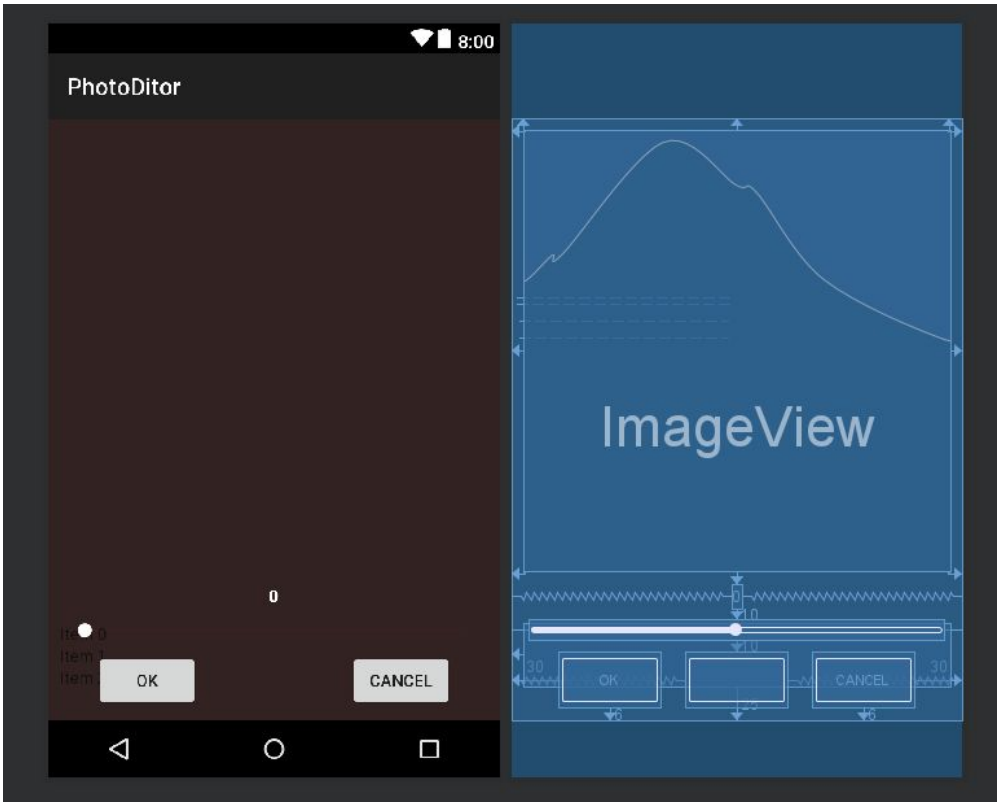
Existeix també el mètode 'invalidateOptionsMenu()'. Aquest mètode invalida el menu actual, i s'utilitza per a notificar algun canvi en l'aspecte que aquest té. És per això que quan es crida aquest mètode el que realment es fa és una crida al mètode anteriorment comentat 'onCreateOptionsMenu()'. Per tant, és en aquest segon mètode on s'haurà de decidir quin arxiu xml es carrega per a inflar el menu.

2.2.3 Android Studio

El concepte de creació d'una aplicació per a Android és bastant fresc, ja que com s'ha explicat anteriorment, és un sistema relativament nou, fet que provoca que les tecnologies per a desenvolupar aplicacions creixi de manera molt ràpida i diferent en cada branca. Durant aquest temps s'ha utilitzat diferents softwares per al desenvolupament d'aplicacions. Inicialment el software oficial era Eclipse. Permet la creació de programes en alguns dels llenguatges més utilitzats avui en dia, tals com Java, C++, JavaScript o PHP. En temes referents a Android, no disposava d'una interfície visual per al desenvolupament de les pantalles que es creen en una app, fet que resultava bastant incomode a l'hora de dissenyar-les i que un dels seus posteriors competidors sí que incorporava.

Al cap d'un temps, aquests va ser substituït per Android Studio en termes de creació d'apps, tot i que actualment es pot continuar utilitzant com a tal. Android Studio ha anat incorporant diferents funcionalitats al llarg de la seva evolució, majoritàriament pensades per a la creació d'aplicacions. Algunes d'aquestes funcionalitats són les següents:

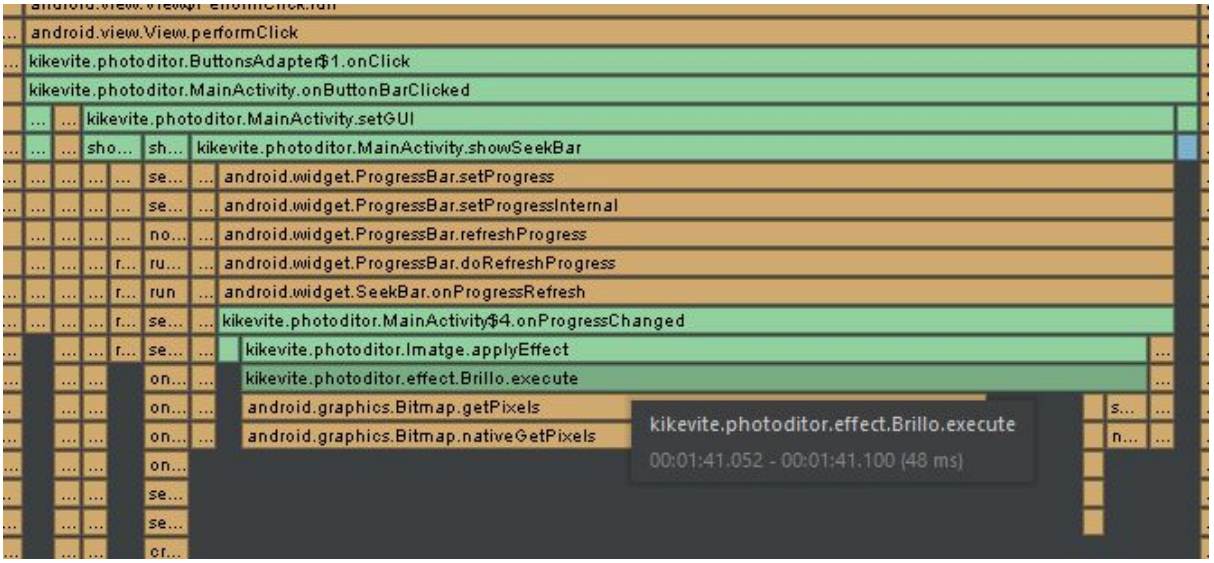
- Creació de layouts de manera visual: facilita als creadors la tasca de col·locar els elements visuals en la posició desitjada dins del layout de la pantalla que visualitzarà l'usuari.



Editor visual de disseny de layouts d'activitats.

- Gradle: incorpora de forma nativa el compilador Gradle, de manera que al instalar el software no cal realitzar altres operacions ni configuracions, a no ser que així es desitgi.

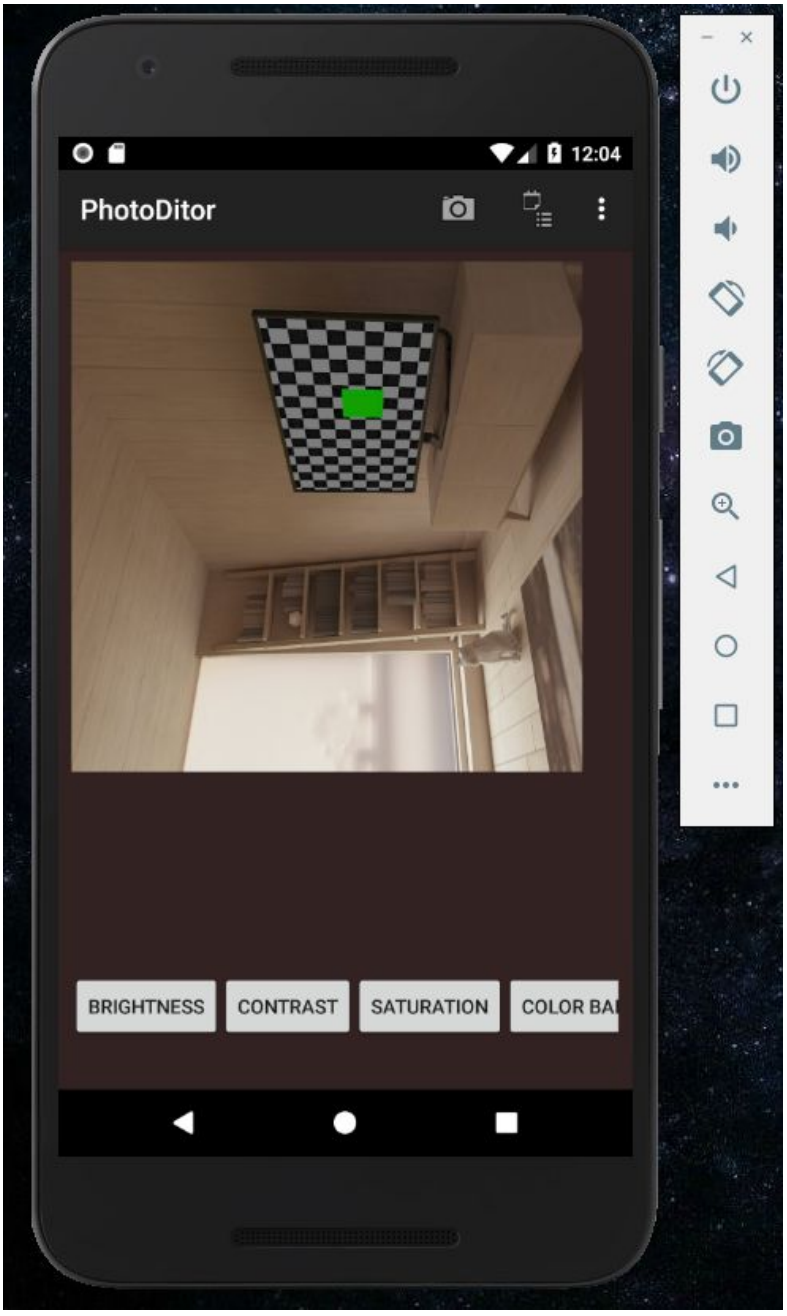
- Android Profiler: una eina d'optimització de codi. Permet veure en temps real el consum de recursos com el processador o de dades d'Internet. També permet visualitzar en quin ordre i durant quant de temps s'està executant cada funció per poder optimitzar-la i detectar possibles errors.



Captura de pantalla de Profiler on es mostra l'ordre de les crides i el temps d'execució.

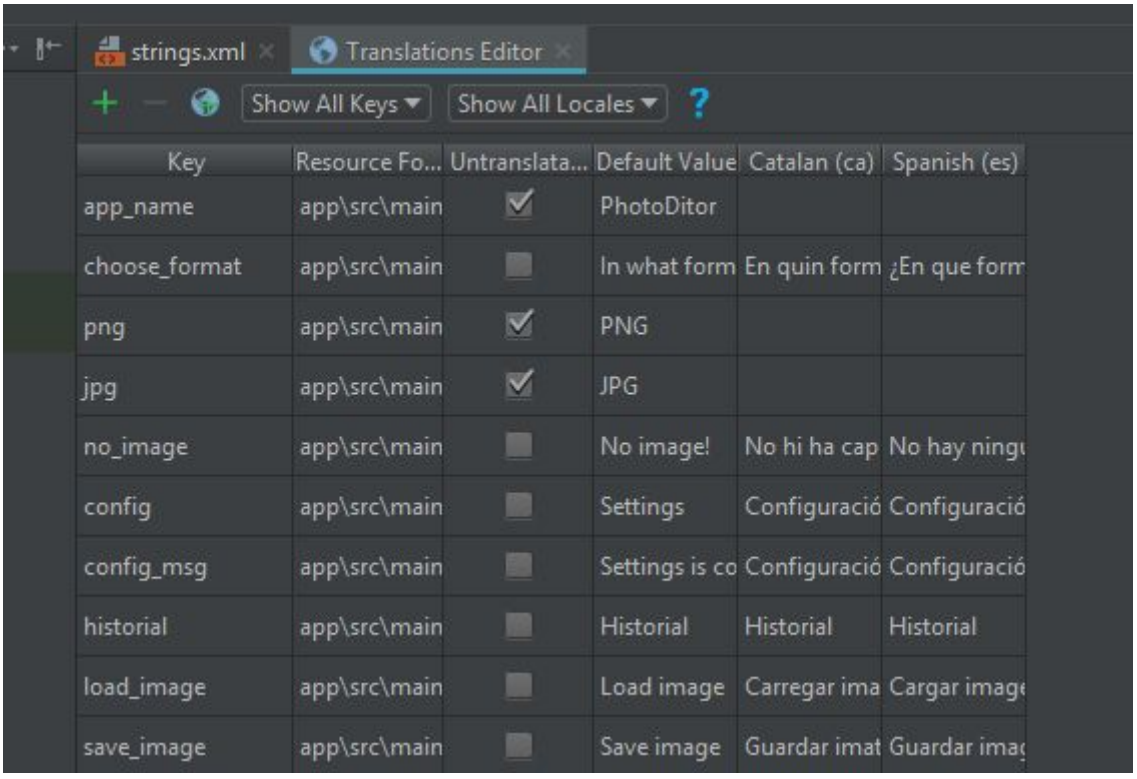
- Assistent de Firebase: permet integrar de forma automàtica les funcionalitats de Firebase, una plataforma de desenvolupament per a mòbils de Google, entre les quals se'n destaca l'emmagatzematge en el núvol per a compartir dades d'usuari, una base de dades en temps real i un servei d'autentificacions d'usuaris amb plataformes creuades (Facebook, Twitter...).

- Android Virtual Device: creació d'un terminal virtual en el qual realitzar proves que elimina la necessitat de disposar d'un smartphone físic. Ofereix la possibilitat de treballar amb totes les versions d'Android que existeixen fins al moment.



Android Virtual Device executant l'aplicació.

- Optimització de traduccions: es disposa d'un gestor de traducció de recursos per tal d'agilitzar la creació de nous recursos de text en diferents idiomes.



Key	Resource Fo...	Untranslata...	Default Value	Catalan (ca)	Spanish (es)
app_name	app\src\main	☑	PhotoDitor		
choose_format	app\src\main	☐	In what form	En quin form	¿En que form
png	app\src\main	☑	PNG		
jpg	app\src\main	☑	JPG		
no_image	app\src\main	☐	No image!	No hi ha cap	No hay ningu
config	app\src\main	☐	Settings	Configuració	Configuració
config_msg	app\src\main	☐	Settings is co	Configuració	Configuració
historial	app\src\main	☐	Historial	Historial	Historial
load_image	app\src\main	☐	Load image	Carregar ima	Cargar image
save_image	app\src\main	☐	Save image	Guardar imat	Guardar imag

Editor de traduccions que incorpora Android.

Altrament a aquest software de desenvolupament, també podem trobar-ne d'altres, com per exemple NetBeans o Processing. Aquest últim, per exemple, proporciona una manera de crear aplicacions de forma més visual, a diferència dels altres.

Finalment també podem trobar altres eines com Xamarin, que forma part de Visual Studio, de Microsoft. Parteix de la idea de crear tan aplicacions per a Android, iOS i Windows Phone reutilitzant el mateix codi, sent el software l'encarregat de realitzar les modificacions necessàries perquè l'aplicació funcioni correctament segons els sistema operatiu final desitjat.

3 Estat de l'art

3.1 Aplicacions semblants

Per al desenvolupament d'aquesta app s'ha dut a terme un treball d'investigació per tal de conèixer quines són les característiques, tant de funcionament com de disseny, de les apps d'edició d'imatges més utilitzades d'avui en dia. És en l'apartat d'edició d'imatges on s'investigarà a fons per poder prendre decisions sobre quins elements ha de tenir la nostra app, quins no i possibles millores que s'hi pot aplicar si així es desitja. Les apps investigades són les llistades a continuació juntament amb l'enllaç corresponent al Play Store de Google:

Afterlight

https://play.google.com/store/apps/details?id=com.fueled.afterlight

Empresa	Afterlight Collective, Inc
Preu	Gratis amb compres integrades (0.73€ per element)
Conté anuncis	No
Nombre de descàrregues	+5.000.000 (a data 04/10/2018)
Puntuació dels usuaris	4.3 sobre 5 amb 52k valoracions
Versió d'Android requerida	4.0.3
Principals permisos necessaris	Ubicació, fotos/multimèdia/arxius, emmagatzematge, càmera, altres

Prisma

https://play.google.com/store/apps/details?id=com.neuralprisma

Empresa	Prisma Labs, inc.
Preu	Gratis amb compres integrades (des de 0.79€ fins a 104.99€)
Conté anuncis	Si
Nombre de descàrregues	+50.000.000 (a data 04/10/2018)
Puntuació dels usuaris	4.2 sobre 5 amb 815k valoracions
Versió d'Android requerida	4.1
Principals permisos necessaris	Fotos/multimèdia/arxius, emmagatzematge, càmera, altres

Snapseed

https://play.google.com/store/apps/details?id=com.niksoftware.snapseed

Empresa	Google LLC
Preu	Gratis sense compres integrades
Conté anuncis	No
Nombre de descàrregues	+50.000.000 (a data 04/10/2018)
Puntuació dels usuaris	4.5 sobre 5 amb 850k valoracions
Versió d'Android requerida	4.4
Principals permisos necessaris	Fotos/multimèdia/arxius, emmagatzematge, càmera, altres

Adobe Photoshop Express

https://play.google.com/store/apps/details?id=com.adobe.psmobile

Empresa	Adobe
Preu	Gratis sense compres integrades
Conté anuncis	No
Nombre de descàrregues	+100.000.000 a data 04/10/2018)
Puntuació dels usuaris	4.2 sobre 5 amb 959k valoracions
Versió d'Android requerida	4.4
Principals permisos necessaris	Identitat, contactes, fotos/multimèdia/arxius, emmagatzematge, informació sobre Wi-Fi, altres

3.2 Anàlisi comparatiu

Tot seguit trobareu un llistat de quins són els punts forts i dèbils de cadascuna de les apps anteriorment esmentades:

Afterlight

- Mostra una icona per indicar a l'usuari que s'està processant la imatge quan s'aplica un efecte.
- Es pot aplicar un mateix efecte moltes vegades, fins a crear un arxiu totalment inservible, però disposa d'un historial que permet tirar endarrere pas per pas.
- L'historial només permet tirar endarrere d'un en un, no permet anar a la imatge original. Per fer-ho, s'ha de descartar les modificacions fetes fins ara a la imatge i tornar a començar.
- Permet triar 3 mides en guardar: petita, mitjana i màxim (no és la mida original de l'arxiu).
- Només permet guardar la imatge en JPG.
- S'observa alguns problemes amb imatges de mides grans, en posar l'app en segon pla i posteriorment tornar-la a utilitzar, en algunes ocasions no es guarden bé les dades amb les quals es treballa i es perden, si s'aplica dues vegades algun efecte, com saturació, per exemple, la imatge es torna negra.
- Disposa d'un menú de configuració; per exemple, es pot personalitzar el color de fons mentre s'edita una imatge

Prisma

- Únicament permet retallar, aplicar un efecte i guardar/compartir la imatge.

Snapseed

- L'historial mostra totes les edicions realitzades, pas per pas i ordenades cronològicament, i permet triar quina versió de la imatge es vol restaurar si així es desitja.
- En guardar una imatge, permet triar el percentatge de compressió si es vol guardar en JPG. També permet guardar en PNG.
- Inclou petits tutorials mostrant quins efectes i en quin ordre s'han d'aplicar per a crear altres efectes fotogràfics més avançats.
- Permet visualitzar les metadades de la imatge, com el número F, velocitat d'obturació o velocitat ISO, si la imatge en disposa.
- No permet capturar imatges amb la càmera del mòbil.

Adobe Photoshop Express

- Incorpora apartats especials que la majoria d'apps no incorpora, com reducció de soroll, reducció de boira...
- Possibilitat de guardar imatges al núvol.
- Permet obrir imatges en format RAW.
- Incorpora un petit editor per a la creació de collages.
- L'historial permet anar endavant i endarrere en els passos d'edició.
- Es pot guardar els ajustos que s'han aplicat com a un sol filtre per aplicar-lo en pròximes edicions.
- Es mostra una petita explicació sobre que fa cada botó la primera vegada que es prem.
- Quan es vol retallar una imatge, permet triar els píxels manualment (1234x1234), i a més té guardats unes quantes mides predefinides (16:9, imatge de perfil de Facebook, Twitter…) i també permet bloquejar la relació d'aspecte per no distorsionar la imatge.
- Incorpora un botó per comparar la imatge original i l'editada.
- Permet decidir el percentatge de compressió JPEG i la mida màxima del fitxer en guardar.
- Hi ha la possibilitat d'editar la imatge amb el "Mode pantalla completa" però no resulta gaire útil si no s'edita una imatge vertical.
- En aplicar un efecte, les barres per controlar el paràmetre d'edició tenen un color de fons per a fer intuir a l'usuari quin serà el resultat si s'augmenta o es disminueix el valor del paràmetre.

Seguidament trobareu diferents taules que comparen quines funcionalitats permet realitzar cadascuna de les anteriors aplicacions, funcionalitats que s'intentarà implementar en la nostra aplicació. En cada taula trobareu el nom de l'aplicació, la funcionalitat que s'ha comprovat, si la compleix amb èxit o no, i una puntuació ponderada. Al final hi haurà una taula amb totes les puntuacions finals, de manera que l'aplicació amb més puntuació serà la qual agafarem com a referència en la implementació i disseny de la nostra.

	Photoshop	Snapseed	Afterlight	Prisma
GENERAL	2	0,5	2	1,5
Pantalla de configuració	Si, bàsica	Si, bàsica	Si	Si, però del social
Obra arxius RAW	Si, però pot fallar	No	No	No
Captura des de càmera	Si	No	Si	Si

Taula comparativa de característiques generals.

	Photoshop	Snapseed	Afterlight	Prisma
EDICIÓ	5	6	4	1
Icona de processant	Si	Si	Si	Si
Aplicar efecte més d'una vegada	No	Si	Si	No
Amb imatges grans falla	No	No	Algun cop	Pagament
Numero d'efectes (no filtres)	17	30 aprox	15	Només filtres
Botó de comparar original i retocada	Si	Si, clic imatge	No	No
Pantalles d'edició personalitzades	Si, algunes	No	Si, algunes	No

Taula comparativa d'edició.

	Photoshop	Snapseed	Afterlight	Prisma
RETALLAR/GIRAR	10,5	8	10	4
Es pot fer amb el dit	Si	Si	Si	Si
Relació d'aspecte lliure	Si	Si	Si	No
Obliga a agafar dins la imatge	Si	Si	Si	Si
Reemplena si la imatge no ompla el quadre	No	No	Si	No
Bloqueig de la relació d'aspecte	Si	Si	Si	Si
Es pot fer introduint valors	Si	No	No	No
Mides predefinides (Facebook, Youtube...)	Si	No	No	No
Relacions d'aspecte predefinides (16:9, 4:1...)	Si	Si	Si	Si
Es mostra la mida mentres es retalla	No, però consultable	No	Si	No
Permet girar amb els dits	No	Si, amb una barra	No	No
Permet girar 90 amb botó	Si	Si	Si	No
Permet girar introduint valors	No	No	No	No
Permet girar amb una barra	Si	Si	Si	No
Permet rotar horitzontal i vertical	Si	Si, horitzontal	Si	No

Taula comparativa de retall/rotació.

	Photoshop	Snapseed	Afterlight	Prisma
HISTORIAL	3	5	3	0
Té historial	Si	Si	Si	No
D'un en un	Si	Si	Si	No
Es pot saltar a qualsevol pas anterior	No	Si	No	No
Mostra totes les edicions que s'han fet	No	Si	No	No
Endavant i endarrere	Si	Si	No	No
Únicament endarrere	No	No	Si	No

Taula comparativa d'historial.

	Photoshop	Snapseed	Afterlight	Prisma
FINALITZAR	5	5	2	1
Permet triar diferents mides al guardar	Si, al inici	Si, a configuració	Si, 3	No
Permet compartir amb altres apps	Si	Si	Si	Si
Només permet guardar en JPEG bàsic	No	No	Si	Si
JPEG personalitzat (%)	Si, al inici	Si	No	No
Guardar en altres formats (PNG...)	No	Si	No	No
Guardar imatges al cloud	Si	No	No	No

Taula comparativa en finalitzar l'edició.

	Photoshop	Snapseed	Afterlight	Prisma
ALTRES	4	3	0	0
Minitutorials d'efectes	No	Si	No	No
Consultar metadades	No	Si	No	No
Editor de collages	Si	No	No	No
Guardar ajustos fets en un filtre per a futures edicions	Si	Si	No	No
Explicació de cada boto la primera vegada que es prem	Si	No	No	No
Mode pantalla completa d'edició	Si	No	No	No

Taula comparativa d'altres característiques.

	Photoshop	Snapseed	Afterlight	Prisma
GENERAL	2	0,5	2	1,5
EDICIÓ	5	6	4	1
RETALLAR/GIRAR	10,5	8	10	4
HISTORIAL	3	5	3	0
FINALITZAR	5	5	2	1
ALTRES	4	3	0	0
RESULTAT FINAL	29,5	27,5	21	7,5

Taula de resultats finals.

Tal com indiquen els resultats, l'aplicació més semblant a la que volem dissenyar és Adobe Photoshop Express. Aquest resultat era l'esperat, ja que la intenció d'aquest projecte era el de crear una aplicació semblant al programa Adobe Photoshop, disponible per a Windows i Mac, i en ser aquesta aplicació desenvolupada pels mateixos creadors d'aquest programa, moltes de les funcions que es vol implementar es troben també en la versió per a mòbils d'aquest famós programa de retoc professional d'imatges.

4 Desenvolupament

En aquesta part del projecte s'ha analitzat quines han estat les necessitats del projecte. S'ha investigat sobre la manera d'implementar els diferents efectes que es vol implementar, com realitzar un historial amb èxit o quins dissenys són els més convenients per a cada pantalla amb les quals l'usuari interactuarà, entre altres. Trobareu les principals idees d'aquest projecte i el seu desenvolupament tot seguit.

4.1 Disseny de l'aplicació

Per començar a desenvolupar l'aplicació, primer de tot s'ha de tenir clar el que es vol fer. S'ha de saber quins tipus d'accions s'ha de realitzar per a poder decidir quins elements s'ha d'utilitzar. És per això que, una vegada decidit que és el que es vol fer, es pot començar a dissenyar les diferents pantalles que tindrà l'aplicació.

Els efectes que de primera es pretenen desenvolupar són els següents:

- Brillantor
- Contrast
- Balanç de blancs
- Corbes
- Equilibri de color
- Zones il·luminades
- Zones d'ombra
- Nivell de blanc
- Nivell de negre
- Saturació
- Filtres d'efectes
- Invertir colors
- Blanc i negre
- Equalitzar
- Canviar resolució
- Canviar mida de la foto (Transformar)
- Retallar la foto
- Rotació d'angle
- Rotació horitzontal
- Rotació vertical
- Capes
- Historial d'edició
- Pinzell
- Text
- Croma
- Format de guardar

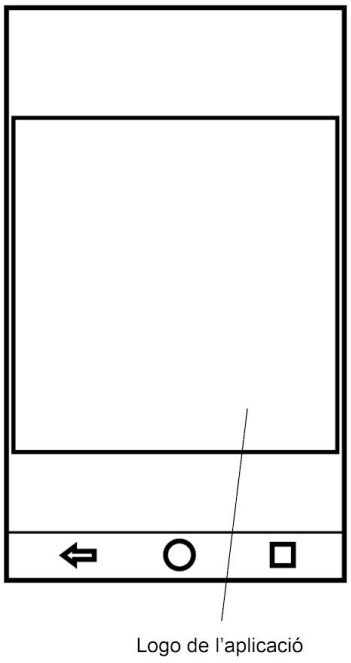
- Moure elements dins la imatge
- Fer zoom a un element
- Eina selecció (quadrat, vareta màgica...)
- Vinyeta
- Afegir gra
- Desenfocament
- Pot de pintura
- Degradat
- Perspectiva
- Tampó de clonar
- Esborrador
- Opacitat

Després d'analitzar les necessitats de l'aplicació, s'ha arribat a la conclusió de què inicialment es necessita un total de tres pantalles diferents. La primera serà una pantalla de benvinguda de tipus 'splash screen'. Aquest tipus de pantalles es mostren durant molt poc temps just després d'executar l'aplicació i en ella es visualitza la icona de l'aplicació, per exemple, per evitar que aparegui directament la primera pantalla que l'usuari utilitzarà realment i crear una transició d'inici més suau. La segona pantalla permetrà escollir la imatge que es vol editar, quin efecte aplicar i editar-lo, i realitzar altres accions, com guardar la imatge o anar a la pantalla d'editar l'historial. La tercera és precisament aquesta, la pantalla d'edició de l'historial, en la qual aquest podrà ser editat. El disseny i la funció de cadascuna d'elles està explicat a continuació.

Dissenys inicial i final pantalla 1 (Splash Screen)

Es mostra quan s'entra a l'aplicació per primera vegada. En aquesta pantalla no es pot realitzar cap acció. Desapareix al cap de poc temps.

Disseny inicial i final "Splash"	Elements	Events	Comportament



Dissenys inicial pantalla 2 (Aplicació d'efectes)

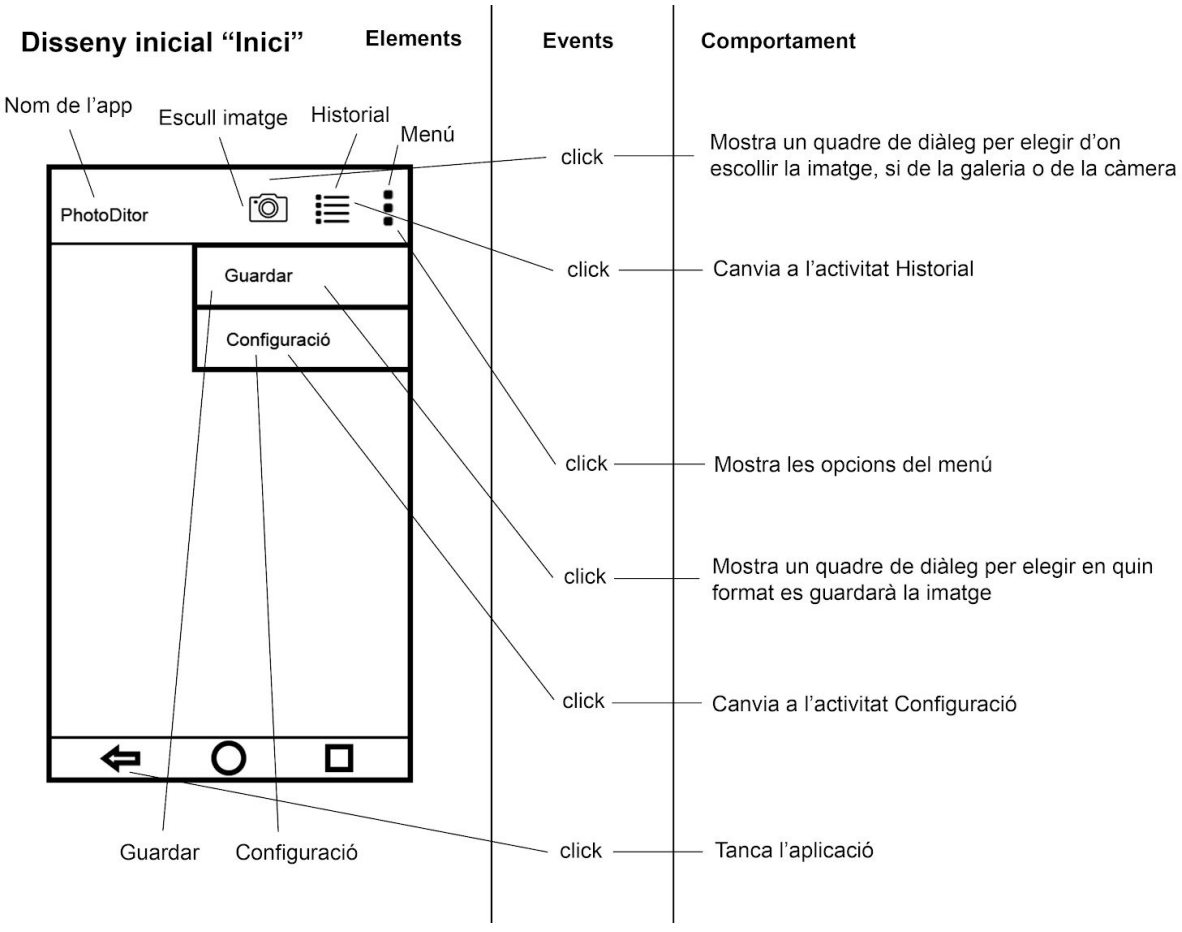
Aquesta serà la pantalla principal de l'aplicació, ja que en ella l'usuari realitzarà la major part de les operacions. Aquestes són les següents:

- Carregar imatges.
- Aplicar efectes.
- Guardar les imatges resultants.
- Consultar informació sobre l'aplicació.

Una vegada s'ha decidit les accions que es duran a terme en aquesta activitat, es procedeix a dissenyar els diferents layouts que es mostrarà a l'usuari. En total són cinc, i es mostren tot seguit.

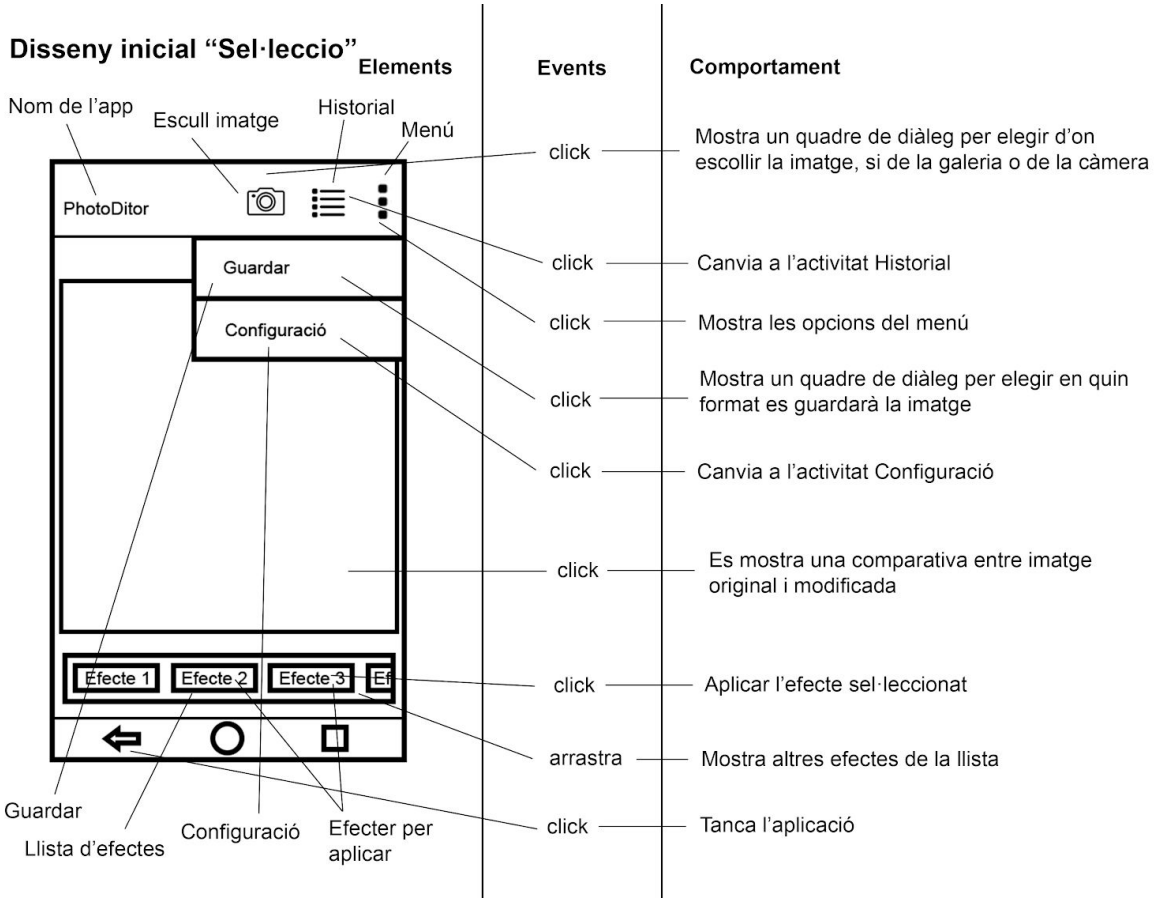
Pantalla inicial

Es mostra després que la pantalla 'Splash Screen' desaparegui de forma automàtica.



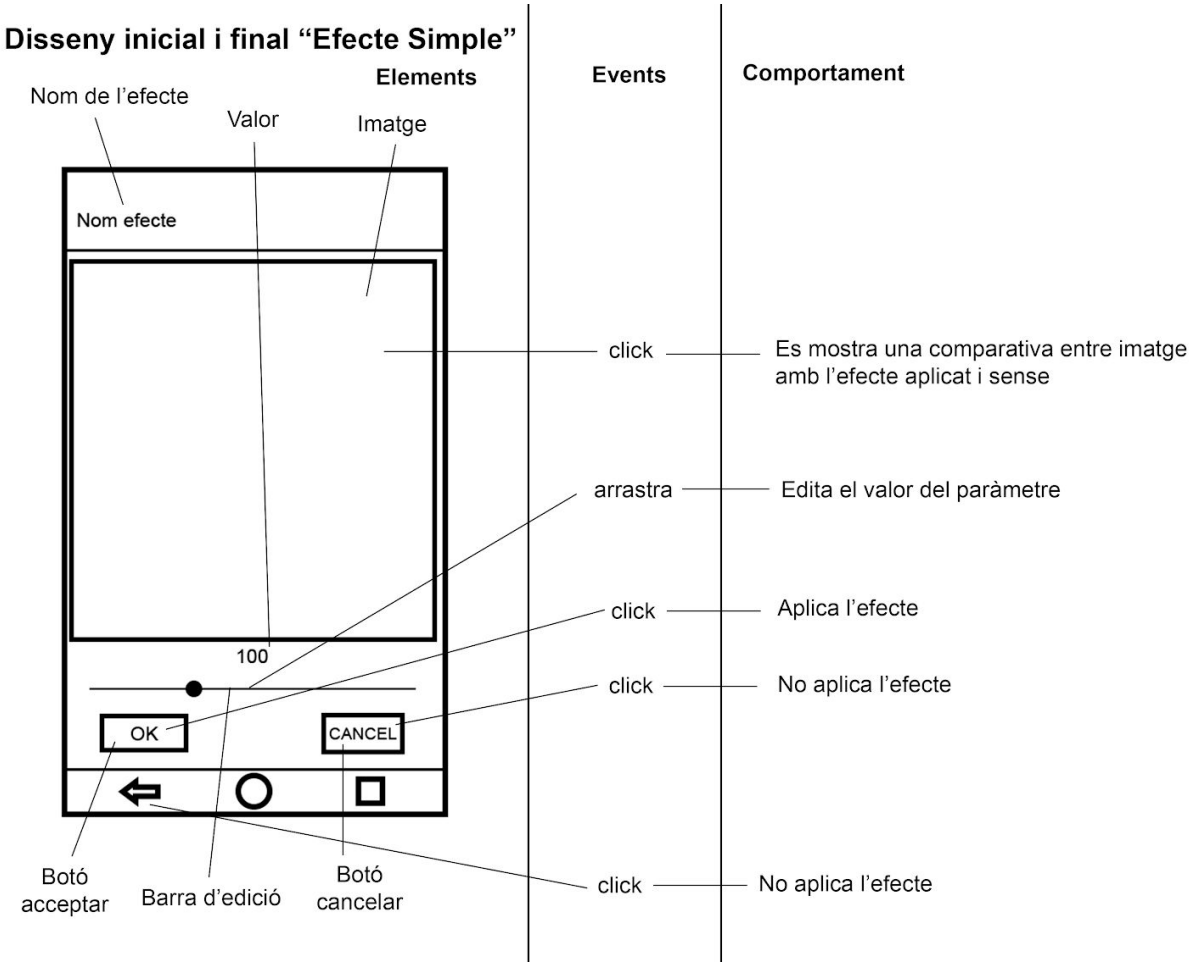
Seleccionar efecte

Una vegada l'usuari ha seleccionat una imatge a través del menú de la pantalla anterior, aquesta es mostra en un requadre situat a la part central de la pantalla. També apareix un element a la part posterior d'aquesta, que és un llistat amb els efectes disponibles a aplicar.



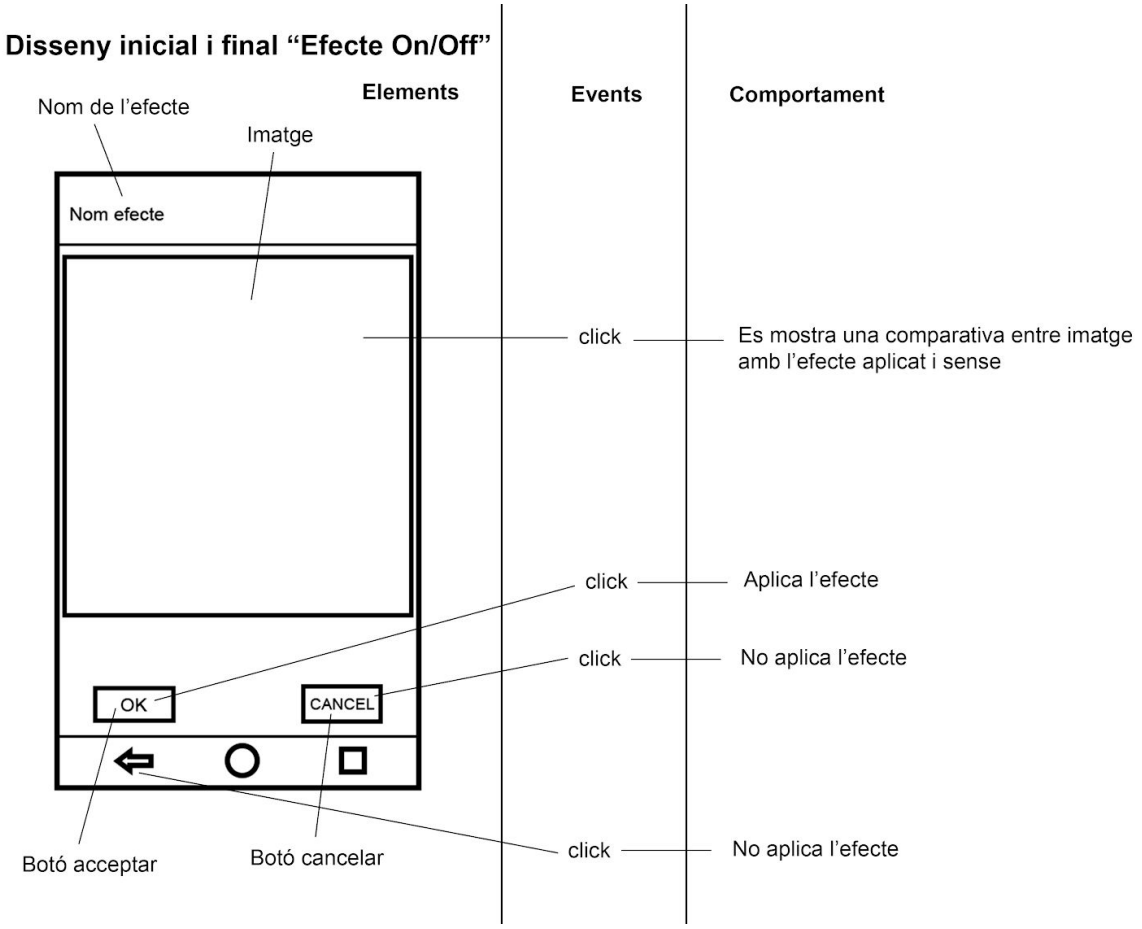
Efecte simple

Es mostra quan s'aplica un efecte simple, que només requereix una barra. Alguns d'ells són el contrast, la saturació o el desenfoc. Trobareu la llista completa en l'apartat **4.2.2.3 Classes** subapartat **Classe GUIValue.**



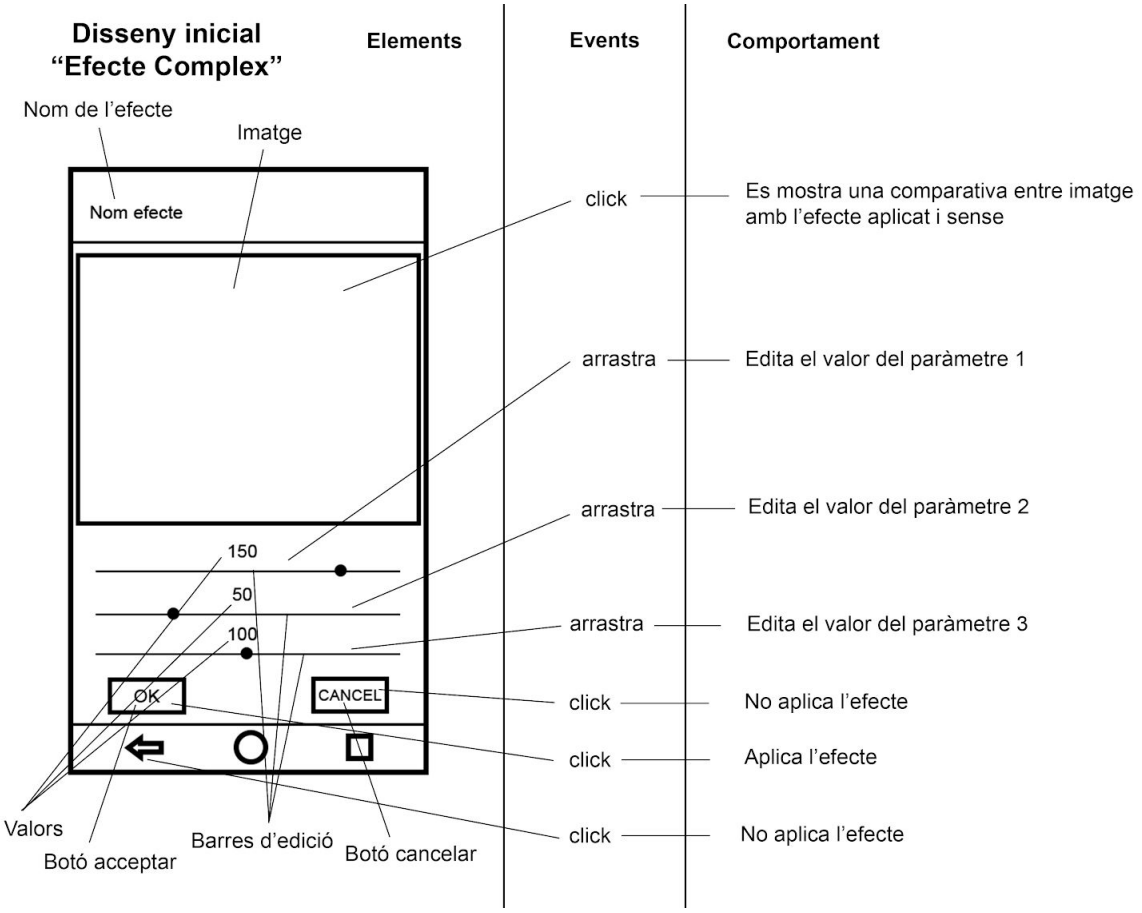
Efecte On/Off

Es mostra quan s'aplica un efecte On/Off, que no requereix cap barra, com per exemple alguns tipus de rotacions. Trobareu la llista completa en l'apartat **4.2.2.3 Classes** subapartat **Classe GUIValue.**



Efecte complex

Es mostra quan s'aplica un efecte complex, que requereix una barra i un botó especial. De moment només s'implementarà un sol efecte d'aquest tipus, equilibri de color.

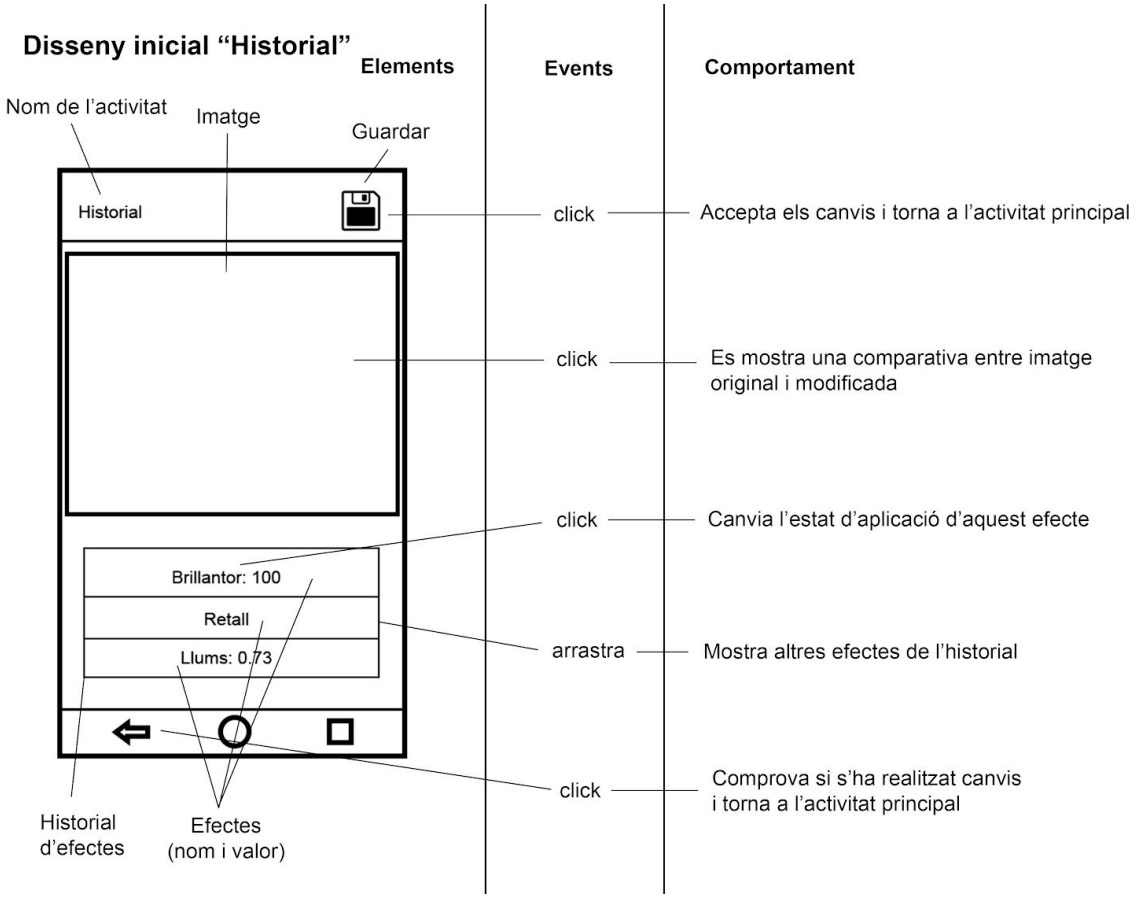


Disseny inicial pantalla 3 (Historial):

Una vegada s'ha aplicat una sèrie d'efectes a la imatge i aquests s'han emmagatzemat en l'historial, a través del menú de la pantalla principal, es pot accedir a la pantalla de modificació de l'historial. Aquesta pantalla tindrà com a finalitat la modificació de l'historial d'edició i s'hi podrà realitzar les següents accions:

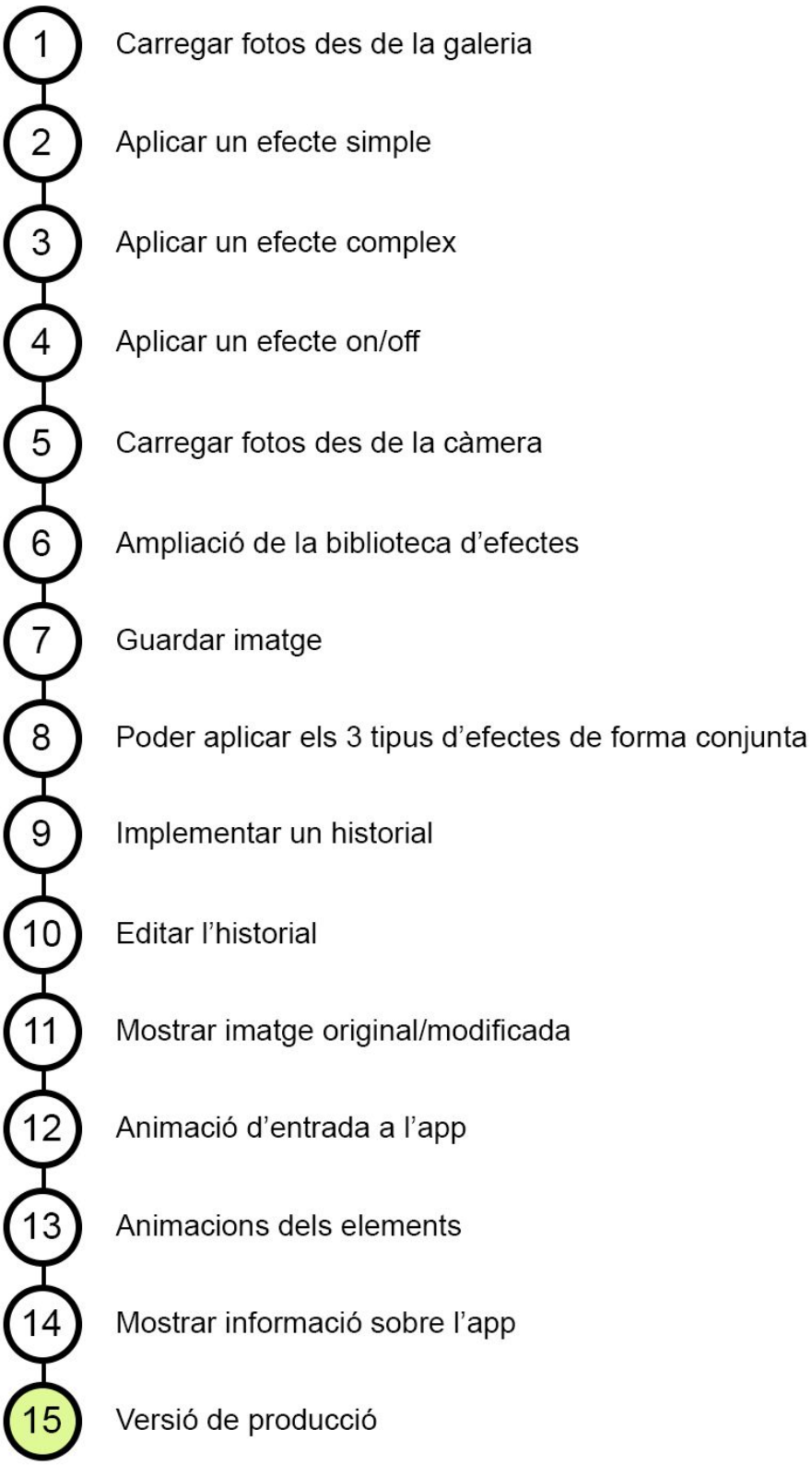
- Visualitzar l'historial.
- Editar l'historial.

Per tal de poder realitzar correctament aquestes accions, s'ha dissenyat el següent esquema d'elements per al layout d'aquesta pantalla:



S'ha de dir que tots aquests dissenys són dissenys inicials, i com a tals, poden patir modificacions durant l'evolució del projecte. Com es veurà en l'apartat de resultats finals, alguns dissenys es veuran modificats i altres no.

Es pretén desenvolupar l'aplicació anant afegint modificacions a cada versió anterior. La planificació inicial de les diferents versions que es crearan i que aniran afegint funcionalitats que s'intentara seguir es mostra en la planificació de versions de l'esquema següent:



1. Carregar fotos des de la galeria: la versió inicial de l'app permet carregar una imatge de la galeria de fotos del smartphone de l'usuari.

2. Aplicar un efecte simple: permet aplicar un efecte que requereix una sola barra per a variar el seu valor, com el contrast.

3. Aplicar un efecte complex: permet aplicar un efecte que necessita tres barres per a variar el seu valor, com l'equilibri de color.

4. Aplicar un efecte on/off: permet aplicar un efecte que només té un valor, com el blanc i negre.

5. Carregar fotos des de la càmera: es pot editar imatges capturades amb la càmera del smartphone.

6. Ampliació de la biblioteca d'efectes: es permet triar entre una varietat d'efectes nous.

7. Guardar imatge: l'usuari pot triar si vol guardar la imatge en format JPG o PNG.

8. Poder aplicar els 3 tipus d'efectes de forma conjunta: es modifica la interfície d'usuari de l'aplicació de manera que segons l'efecte que s'ha elegit es mostrin els elements necessaris per a interactuar correctament amb ell.

9. Implementar un historial: es crea un historial on s'emmagatzema tots els efectes que l'usuari ha aplicat a la imatge.

10. Editar l'historial: es pot visualitzar els efectes que hi ha guardats a l'historial i decidir quins es vol mantenir i quins esborrar.

11. Mostrar imatge original/modificada: quan es fa clic sobre la imatge, es mostra la versió original, si no s'està aplicant cap efecte, i la versió sense l'efecte, si se n'està aplicant un.

12. Animació d'entrada a l'app (Splash screen): en entrar a l'aplicació es mostra el logo de l'aplicació en una animació.

13. Animacions dels elements: quan es fa clic en certs elements, desapareixen d'una manera gradual, donant fluïdesa a les transicions.

14. Mostrar informació sobre l'app: es pot consultar la versió de l'app i dades de contacte amb el desenvolupador.

15. Versió de producció: primera versió apta per a ser publicada a la Play Store.

4.2 Implementació de l'aplicació

Una vegada s'ha dissenyat les diferents pantalles que es vol mostrar als usuaris de l'aplicació, es procedeix a la implementació de les funcionalitats que es vol dur a terme. És en aquest apartat on es transformen els efectes que es detallen en l'apartat 2. Background en el codi que s'executara i transformarà les imatges.

4.2.1 Eines utilitzades

Per tal d'implementar el codi s'ha utilitzat el següent programari. Ambdós són àmpliament reconeguts dins del món de la programació, fet que proporciona una gran varietat d'exemples per a l'aprenentatge i resolució de dubtes i problemes que poden sorgir durant la implementació del projecte. Aquests són Android Studio i Git, combinat amb el website GitHub.

Android Studio

Com s'ha deixat entreveure en l'apartat 2. Background, el software de desenvolupament que s'utilitzarà en aquest projecte és Android Studio. És un entorn de desenvolupament molt potent que ofereix moltíssimes possibilitats i està altament pensat per a la creació d'aplicacions. També incorpora de manera nativa altres funcionalitats que no estan directament relacionades amb el desenvolupament d'aplicacions, però que faciliten molt les tasques a realitzar. En el següent apartat se n'explica una d'elles que s'utilitzarà bastant durant aquest projecte.

Git

En aquest tipus de projectes és habitual treballar en equip, i és per això que s'utilitzen eines per a crear diferents branques de treball. A més a més, s'acostuma a tenir un control sobre les diferents versions que es van desenvolupant, fet que permet tornar endarrere en el cas que algun aspecte del projecte no estigui ben desenvolupat i no vulgui ser utilitzat.
Per a poder fer això s'utilitza el software de control de versions Git, que permet realitzar les accions anteriorment comentades.

Git és tan sols un dels diferents softwares de control de versions que existeix, però avui en dia és un dels més utilitzats i, per tant, el que té més documentació i exemples els quals poder analitzar el seu funcionament.
És el software que s'utilitza en aquest projecte pel fet que ja s'ha utilitzat en projectes anteriors i es coneix el seu funcionament, fet que permet estalviar temps en formació sobre aquest aspecte i que podrà ser dedicat a altres tasques.

Per la part de l'allotjament d'aquest projecte de git, s'utilitza Github.com, un website que permet crear un repositori on podrem penjar els arxius del nostre projecte per tal de compartir-ho amb tothom i des de la qual qualsevol altra persona, amb els suficients permisos, pot modificar el projecte. Això permet una transferència d'arxius de forma més fluida i fàcil, tot i que aquest projecte ha estat desenvolupat individualment.

Aquestes dues eines mencionades són d'utilització gratuïta.

4.2.2 Implementació

Com s'ha comentat en anteriors apartats, l'estructura d'un projecte d'aplicació Android està format per diferents elements que obligatòriament has d'estar-hi perquè l'aplicació tingui un correcte funcionament. En aquest apartat hi trobareu una explicació dels arxius que s'ha creat durant el desenvolupament.

4.2.2.1 Activitats

En apartats anteriors ja s'ha descrit que és una activitat. En aquest, ens centrarem a explicar quines són les activitats que formen la nostra aplicació i com han estat desenvolupades per tal de complir els objectius assignats a cadascuna d'elles.

SplashActivity

L'activitat SplashActivity és una activitat que té la funcionalitat d'una splash screen. Una splash screen és una pantalla de benvinguda a l'aplicació. En ella normalment es mostra la icona de l'app o alguna imatge relacionada amb l'app en una animació de manera que l'entrada de l'usuari a l'aplicació resulti més agradable.

En aquest cas s'ha decidit aplicar una animació que està formada per les següents fases:
Primer apareix una activitat en la qual es mostra gairebé a pantalla completa (exceptuant la barra d'estat i la barra de navegació) la icona que s'ha creat per aquesta app sobre un fons de color gris mig.

Captura de pantalla de SplashActivity.

Passats 1,2 segons, aquesta activitat comença a desaparèixer mitjançant una dissolució, és a dir, es baixa l'opacitat d'aquesta fins a arribar al 0%, per donar pas a l'activitat principal MainActivity.

Al mateix temps que aquesta activitat desapareix, s'aplica també l'animació contrària a aquesta a l'activitat que apareix, és a dir, s'augmenta l'opacitat d'aquesta fins a arribar al 100%. D'aquesta manera s'aconsegueix donar un efecte de transició suau en el canvi entre les dues activitats.

Per a poder dur a terme aquesta animació, primerament s'ha de crear els dos arxius XML en els quals hi haurà el contingut de l'animació en la carpeta d'animacions dels recursos de l'aplicació. Aquesta és la carpeta 'anim' que es troba dins la carpeta 'res'. Aquests dos arxius tenen el següent codi:

Arxiu 'fade_in.xml':

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
   android:duration="1000"
   android:fromAlpha="0.0"
   android:interpolator="@android:anim/accelerate_interpolator"
   android:toAlpha="1.0" />
```

Arxiu 'fade_out.xml':

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
   android:duration="1000"
   android:fillAfter="true"
   android:fromAlpha="1.0"
   android:interpolator="@android:anim/accelerate_interpolator"
   android:toAlpha="0.0" />
```

Es pot observar que els dos arxius tenen alguns dels següents atributs:

- duration: temps que dura l'animació.
- fromAlpha: opacitat inicial del View al qual s'aplica l'animació.
- interpolator: tipus de progressió que s'aplica a l'animació.
- toAlpha: opacitat final del View al qual s'aplica l'animació.
- fillAfter: determina si la transformació final s'aplica una vegada l'animació ha finalitzat.

Amb aquests 5 simples atributs tenim les suficients eines per a poder realitzar l'animació que s'ha descrit anteriorment.

Una vegada s'han creat aquests arxius, es procedeix amb la creació d'una activitat, que anomenarem SplashActivity. En aquesta activitat s'implementarà el codi que s'acostuma a utilitzar per a canviar entre activitats, però la diferència recau en el fet que se li aplicarà una transició, que serà la descrita en els dos arxius XML.

Crearem un atribut estàtic i constant dins l'activitat. En aquest li assignarem el valor de la duració de la transició en mil·lisegons. En el nostre cas 1,2 segons equival a 1200 mil·lisegons:

```
private static final int SPLASH_DISPLAY_LENGTH = 1200;
```

Seguidament, dins del mètode 'onCreate()' i després de la crida al seu 'super()' es crearà un Handler, que serà on s'implementarà el canvi d'activitats i la transició entre elles:

```
new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
                startActivity(new Intent(SplashActivity.this, MainActivity.class));
                overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
                finish();
        }
}, SPLASH_DISPLAY_LENGTH);
```

A aquest Handler se li aplicarà el mètode postDelayed(), que executa les ordres que continguin el mètode 'run()' de l'interface Runnable que es passa com a primer paràmetre un cop hagi transcorregut el temps indicat en el segon paràmetre.
Es crearà una interface Runnable que només tindrà una execució, i en ella s'implementarà el mètode 'run()'.
En aquesta part es cridarà al mètode 'startActivity()' passant com a paràmetre un Intent explícit que rep com a paràmetres en el seu constructor l'activitat que es deixa endarrere i la 'class' de l'activitat a la qual es pretén anar.
Tot seguit, se sobreescriu la transició que té assignada per defecte aquest canvi d'activitats. Per fer-ho, s'utilitza el mètode 'overridePendingTransition()', el qual rep com a paràmetres les referències als dos arxius XML que s'ha creat anteriorment. Una vegada realitzada aquesta operació, es procedeix a donar fi a l'activitat que s'abandona cridant el mètode 'finish()', que mata l'activitat.

MainActivity

La MainActivity és l'activitat principal de l'aplicació. És on es du a terme tota la part del processament que s'aplica a les imatges, i això la converteix en la principal i més important. Conté diferents pantalles amb diferents layouts, tal com s'ha vist en la secció de disseny. Aquests són:

- Layout d'inici: corresponent amb el disseny "Inici".
- Layout de selecció d'efecte: corresponent amb el disseny "Selecció".
- Layout d'efecte simple: corresponent amb el disseny "Efecte Simple".
- Layout d'efecte complex: corresponent amb el disseny "Efecte Complex".
- Layout d'efecte on/off: corresponent amb el disseny "Efecte On/Off".

Si observem els dissenys inicials, podem fer-nos una idea sobre el tipus d'elements necessaris en aquesta activitat i, després d'analitzar les seves funcions, es conclou que el tipus més adequat d'elements d'Android a utilitzar és el següent:

- Menú superior: per elegir entre les possibles accions que es poden realitzar.
- ImageView: on es mostra la imatge que s'edita i els resultats d'aplicar-hi els efectes.
- RecyclerView: per a elegir quin efecte s'aplicarà a la imatge.
- TextView de valor: per mostrar el valor de la SeekBar
- SeekBar d'edició: per variar el valor de l'efecte que s'està aplicant.
- Button acceptar: per acceptar l'efecte que s'ha aplicat.
- Button cancel·lar: per cancel·lar l'efecte que s'ha aplicat.
- Button d'opció: per seleccionar quin dels tres valors de l'efecte 'equilibri de color' s'està modificant.

Es crearà un arxiu xml de layout en el que s'afegiran aquests elements de manera conjunta, ja que alguns elements com els Buttons desenvolupen les mateixes tasques en vàries pantalles.

Abans d'entrar en detall, es llistarà quin son els atributs i mètodes d'aquesta activitat:

Atributs:

- private static final int galleryRequestCode: codi de retorn de l'activitat galeria.
- private static final int cameraRequestCode: codi de retorn de l'activitat càmera.
- private static final int goHistorialActivityRequestCode: codi de retorn de l'activitat 'HistorialActivity'.
- public static String KEY_ID: variable per transmetre els id's dels efectes de l'historial entre activitats.
- public static String KEY_VALUE: variable per transmetre els valors dels efectes de l'historial entre activitats.
- private static final int MY_PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE: variable necessària per a la petició de permisos d'escriptura.
- private static final int MY_PERMISSIONS_REQUEST_CAMERA: variable necessària per a la petició de permisos de càmera.

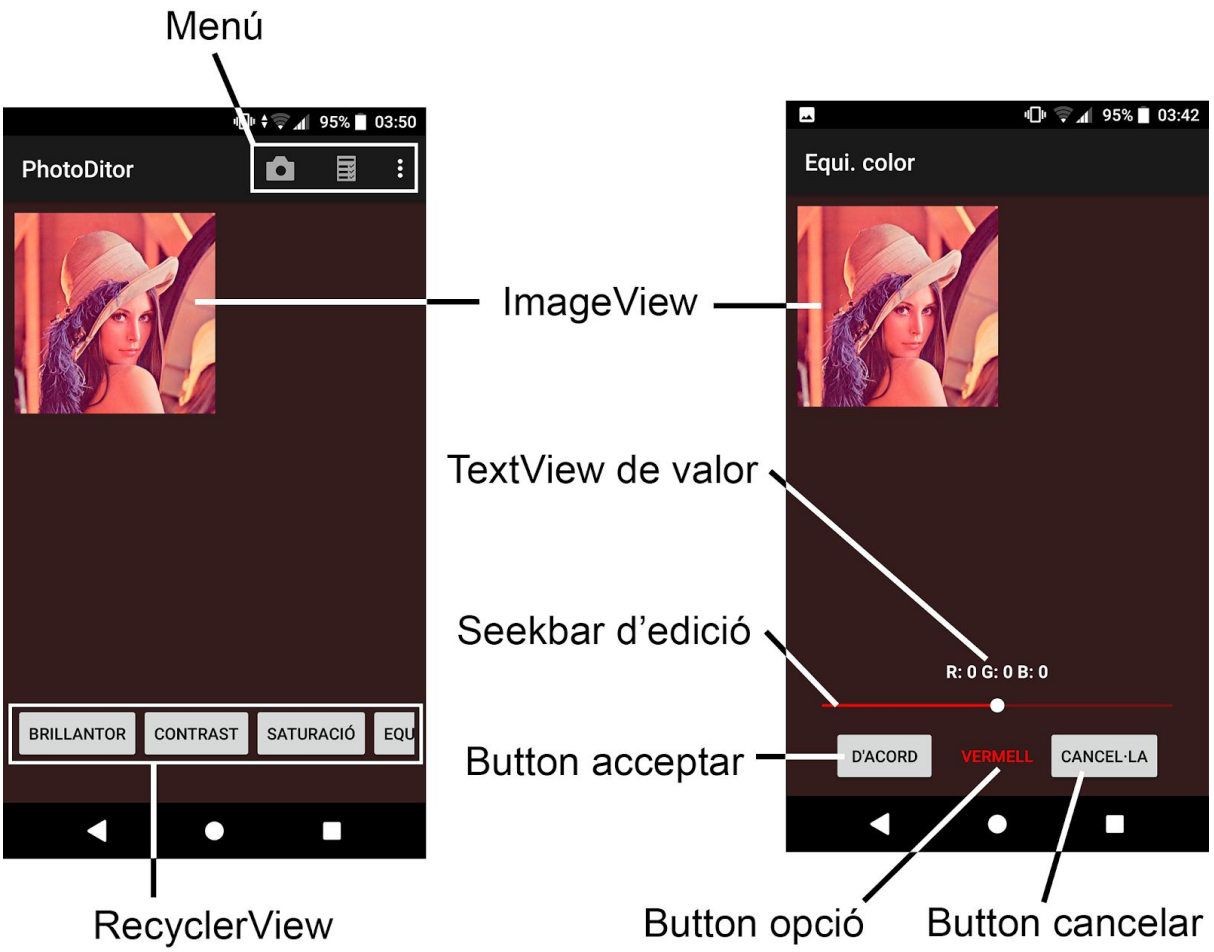
- private boolean cameraAndWrite: per saber si es realitza la petició dels dos permisos de forma conjunta o separada.
- private View cropPoint1, cropPoint11, cropPoint2, cropPoint3, cropPoint4, cropArea: elements per a formar l'àrea de retall de l'efecte retallar, que es veurà més tard.
- private ImageView imageView: on es mostrarà la imatge.
- private TextView numText: on es mostrarà el valor de l'efecte.
- private SeekBar bar: barra per editar el valor de l'efecte.
- private Button btnAccept, btnCancel, btnBalance: botons d'acceptar, cancel·lar i d'opció en efectes complexos.
- private RecyclerView myRecyclerView: per mostrar la llista d'efectes disponibles.
- private Imatge ima: imatge que s'edita. Els detalls sobre la classe es troben en futurs apartats.
- private ArrayList<Effect> historial: historial d'efectes.
- private Effect efecte: efecte que s'aplica.
- private String mCurrentPhotoPath: arxiu de la foto que s'ha realitzat.
- private Uri photoURI: Uri on es guardarà la foto quan es realitzi.
- private int rgb: per controlar el text i el color d'aquest, que apareix al BalanceButton.
- private int screenWidth: ample de la pantalla, per la posició del NumText.
- private final int[] tempEffectValue: valor necessari per emmagatzemar el valor d'un efecte temporalment.
- private PointF p1 = new PointF(), p2 = new PointF(), p3 = new PointF(), p4 = new PointF(): punts per emmagatzemar les coordenades inicials de les cantonades de l'àrea de retall
- private boolean moveAreaOrPoint: per saber si es mou l'àrea de retall o una de les seves cantonades.
- private int ample, alt: ample i alt de l'àrea de retall.
- private float dX, dY: diferències de desplaçament en fer clic a la pantalla per a moure l'àrea de retall.
- private int pointToMove: punt de l'àrea de retall que es mou.
- private int tamanyLineaCrop: amplada de la línia que limita l'àrea de retall.
- private int mShortAnimationDuration: temps de duració d'una animació.

Mètodes:

- private boolean moveAreaOrPoint(MotionEvent event): determina si, segons la zona on s'ha realitzat el clic, s'ha de moure l'àrea de retall o una de les cantonades d'aquesta.
- private int getPointToMove(MotionEvent event): determina quina de les cantonades de l'àrea de treball s'ha de moure a partir de l'event de clic.
- private void getWritePermise(): sol·licita permisos d'escriptura.
- private void getCameraPermise(): sol·licita permisos de càmera.
- private void chooseImageFromGallery(): escull imatge de la galeria.
- private void takeImageFromCamera(): captura imatge de la càmera.
- private File createImageFile(): crea el fitxer de la imatge capturada amb la càmera.
- private String createPhotoName(): crea un nom per a la fotografia que s'ha captat.
- private String[] getEffectsNames(): obté el nom de tots els efectes dels recursos.
- private void recyclerViewConfiguration(): configuració del RecyclerView.

- private void setGUI(int i): mostra cert tipus de layout.
- private void show...(): mostra un View determinat, aplicant-hi l'animació que es comentarà més endavant. Hi ha una funció per cada View de la pantalla.
- private void hide...(): oculta un View determinat. Hi ha una funció per cada View de la pantalla.
- private void addEffectToHistorial(): afegeix un efecte a l'historial i aplica l'efecte, a no ser que aquest es trobi situat en el seu valor per defecte.
- private void cancelAddEffectToHistorial(): anul·la l'addicióaddició de l'efecte a l'historial i cancel·la l'efecte.
- public ArrayList<Integer> getHistorialIds(): obté una llista dels id's dels efectes de l'historial.
- public ArrayList<Integer> getHistorialValues(): obté una llista dels valors dels efectes de l'historial.
- private void saveFinalImatge(): pregunta en quin format es vol guardar la foto i la guarda.
- public void resetData(Uri uriFromImage): reseteja totes les dades i assigna la imatge de la uri passada com a nova imatge a editar.

Tot seguit es detalla el funcionament dels elements de MainActivity. Es comença explicant en detall les funcions que executa cada element dels anteriorment nomenat i s'acabarà explicant la resta de funcions que trobem en aquesta activitat.



Elements de l'activitat principal.

Menú

En aquest menú es pot realitzar un total de 5 accions:

- Carregar una imatge.
- Anar a la pantalla d'edició de l'historial.
- Guardar la imatge final.
- Anar a la pantalla de configuració.
- Visualitzar informació sobre l'aplicació.

S'ha de tenir en compte que únicament es vol mostrar el menú quan no s'està editant un efecte, és per això que es canviarà la visibilitat del menú quan se n'estigui aplicant un. Això es farà utilitzant el mètode 'invalidateOptionsMenu()'.

En la primera opció s'escull d'on es vol agafar la imatge. Es dóna a escollir entre la galeria o la càmera. Si s'escull la càmera, es comprova si es té permisos i, si no se'n té, se'n demana mitjançant el mètode 'getCameraPermises()' i 'getWritePermises()', que demanen els respectius permisos d'escriptura de fitxers i d'accés a la càmera, ja que són necessaris per a poder realitzar fotografies. El procés de petició de permisos s'explicarà més endavant.
En els dos casos, en retornar de la galeria o de la càmera, es mostra la imatge escollida en l'ImageView. Si no s'escull cap imatge la pantalla no es modifica.

En la segona opció del menu es canvia d'activitat, iniciant 'HistorialActivity', per a editar l'historial. Aquesta activitat serà explicada posteriorment.

La tercera opció serveix per guardar la imatge final. Es comprova si es té permís d'escriptura i, si no se'n té, se'n demanen, i si se'n té, es procedeix a guardar la imatge a través del mètode 'saveFinalImatge()'.

La quarta opció està preparada per tal de permetre l'accés a l'activitat de configuració. Aquesta activitat de moment no ha estat creada, ja que inicialment es pretenia desenvolupar funcionalitats que necessitaven aquest requisit però no s'han realitzat. De moment s'ha creat aquesta opció per si en un futur es necessita.

La cinquena i última opció permet mostrar informació sobre l'aplicació. En concret es mostra la versió de l'aplicació i un correu electrònic de contacte.

ImageView

És l'element que mostra la imatge que s'edita. Per defecte mostra la imatge amb tots els efectes aplicats. Si es clica l'element en el moment que no s'està aplicant un efecte es mostra la imatge inicial, per a permetre fer comparacions. Si es clica quan s'està aplicant un efecte, es realitza una comparativa entre la imatge amb aquest efecte aplicat amb la imatge sense l'efecte aplicat.

RecyclerView

Es mostra el llistat d'efectes disponibles a aplicar. En arrossegar el dit de forma horitzontal es mostren tots els efectes disponibles fins a arribar al final. Cada element d'aquest llistat està format per un Button que, al ser clicat, realitza les següents accions:

- Genera un efecte del tipus seleccionat.
- Canvia el nom que es mostra a la barra d'aplicació i es mostra el nom de l'efecte.
- Desactiva el menu.
- Configura el layout a mostrar segons el tipus d'efecte.

Per aquest element s'ha creat un adaptador anomenat 'ButtonsAdapter', seguint les indicacions d'apartats anteriors. Finalment, el RecyclerView està configurat per tal que el llistat es mostri en posició horitzontal i sense cap mena de separador d'elements, amb una mida fixa.

TextView de valor

En el TextView [22] es mostra quin és el valor d'aplicació de l'efecte sobre la imatge. Únicament realitza una funció informativa. Es modifica la posició d'aquest perquè es mostri sempre sobre de la posició de la barra d'edició, explicada a continuació.

SeekBar d'edició

Aquesta barra permet a l'usuari editar el valor d'aplicació de l'efecte. L'usuari desplaça l'element visual anomenat thumb cap a la dreta o l'esquerra de la pantalla per a augmentar el valor o disminuir-lo. Quan encara no s'ha realitzat cap modificació, aquest es troba situat en el valor per defecte d'aplicació de l'efecte. També es realitza el càlcul de la posició en què s'ha de situar el TextView de valor. A aquesta barra se li ha d'assignar un valor màxim, que es correspon al valor Effect.MAX_BAR_VALUE, que s'explicarà en l'apartat 'Classe Effect'.

Button acceptar

Amb aquest botó s'aplica un efecte a la imatge. Per tant, s'ha d'afegir l'efecte aplicat a l'historial i, com l'aplicació de l'efecte ja s'ha finalitzat, es torna a mostrar el RecyclerView per a la selecció d'un nou efecte, si així es desitja. També es canvia el text on es mostrava el nom de l'efecte per el nom de l'aplicació.

Button cancelar

Contràriament a l'anterior botó, aquest restaura la imatge a l'estat en què es trobava abans d'aplicar l'efecte. També canvia el nom de la barra d'aplicació i restaura el layout mostrant l'adequat per a seleccionar un efecte.

Button opció

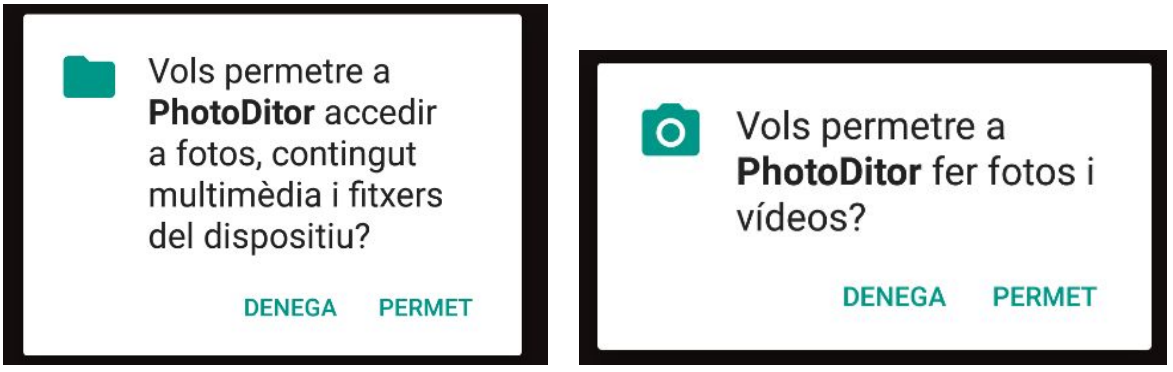
S'utilitza només en l'efecte 'Equilibri de color', i s'encarrega de canviar quina de les tres components de color es modificarà en variar el valor d'aplicació amb la SeekBar. Quan es prem, a part de realitzar la funció anterior, es modifiquen el seu text informatiu i el color d'aquest per tal que l'usuari tingui clar quina component es modificarà.

A part d'aquests elements, també es mostra per defecte la barra de navegació. Aquesta barra és nativa d'Android i està situada en la part inferior de la pantalla. Està formada per tres botons, que permeten a l'usuari tirar endarrere (sortir de l'aplicació, si s'escau), tornar a la pantalla inicial d'Android, i mostrar les aplicacions que s'estan executant actualment. En aquesta activitat s'ha substituït la implementació del botó de sortir de l'aplicació de tal manera que, en un estat diferent del d'aplicar un efecte, es finalitza l'aplicació, però si s'està aplicant un efecte, es realitzin les mateixes accions que si l'usuari hagues clicat el botó de cancel·lar. Això es realitza sobreescrivint el mètode 'onBackPressed()'.

Per al correcte funcionament d'aquesta aplicació ha estat necessària la implementació de codi que permeti a l'usuari concedir els permisos d'usuari [23] necessaris. Aquests han estat el d'escriptura sobre la memòria externa i el d'utilització de la càmera incorporada del dispositiu, ambdós de tipus perillós.

Permisos d'usuari

Un smartphone és un dispositiu personal, i com a tal, conté informació privada sobre l'usuari. És per això que Android aplica una capa de protecció sobre aquest tipus d'informació, implementant els anomenats "permisos".



Exemples de missatges de petició de permisos.

Els permisos es poden concedir en diferents moments:

- Petició de permisos en temps d'execució: són els que es demanen un cop l'aplicació ja està instal·lada. Es mostra un quadre de diàleg en el qual l'usuari pot decidir si concedir el permís o no. També es disposa de l'opció de "No tornar a preguntar" que fa que si l'usuari denega el permís, aquesta petició no es torni a realitzar la pròxima

vegada que es necessiti. Si aquesta opció no es marca, es preguntarà cada vegada que es requereixi el permís fins que aquest s'accepti.

- Petició de permisos durant la instal·lació: són els que, com el seu nom indica, es demanen durant la instal·lació de l'aplicació al dispositiu. S'acostuma a utilitzar aquest mètode quan l'aplicació requereix obligatòriament els permisos necessaris. Si no es concedeixen els permisos, no es procedeix amb la instal·lació. També es diferencien en el fet que es realitza una única petició en la qual s'accepten tots els permisos de cop, en lloc d'un per un, agilitzant així el procés.

Si tenim en compte el nivell de protecció dels permisos, podem trobar:

- Permisos normals: suposen un nivell de risc baix. Si una aplicació sol·licita un permís d'aquest nivell, Android li concedeix de manera automàtica, ja que es considera que no suposa cap perill per a la seguretat de les dades de l'usuari. Poden ser tals com la consulta de l'hora, utilització d'internet, Bluetooth o la vibració.

- Permisos perillosos: requereixen l'aprovació explícita de l'usuari. A diferència dels normals, aquests sí que impliquen la utilització de les dades personals de l'usuari, com missatges de text, contactes de telèfon, gravacions de micròfon…

- Permisos especials: únicament formen part d'aquest grup dos permisos. SYSTEM_ALERT_WINDOW i WRITE_SETTINGS. El primer serveix per a la creació de pantalles que es mostren per sobre de qualsevol altra app, com per exemple les notificacions. La segona consisteix en la modificació de la configuració del sistema operatiu. Pels interessos d'aquesta aplicació i la no utilització d'aquests, no s'entrarà en més detalls.

Per a la correcta petició d'un permís, s'ha d'incloure aquest en el manifest de l'app de forma obligatòria, encara que l'usuari pugui denegar el permís. S'utilitzarà la següent nomenclatura:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.android.app.myapp" >
   <uses-permission android:name="android.permission.RECEIVE_SMS" />
</manifest>
```

En el cas dels permisos normals no és necessari implementar un mètode en el codi per a obtenir el permís, sinó que Android el concedeix de forma autònoma. En canvi, els permisos perillosos sí que requereixen la implementació de funcions. Tot seguit es mostra l'exemple de petició del permís d'utilització de la càmera:

```
if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.CAMERA)) {
                // Demana permis de camera opcio 1
        } else {
                // Demana permis de camera opcio 1
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.CAMERA}, MY_PERMISSIONS_REQUEST_CAMERA);
        }
}
```

Després de realitzar la petició del permís, s'ha de controlar la resposta de l'usuari, ja que pot acceptar o denegar aquesta petició. Això s'aconsegueix sobreescrivint el mètode onRequestPermissionsResult():

```
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[]
grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_CAMERA: {
                if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    // Permis camera concedit
                } else {
                    // Permis camera denegat
                    }
                return;
            }
        }
}
```

Tot i això, s'ha de tenir en compte que l'usuari pot concedir un permís i més tard decidir revocar-lo des de la configuració del sistema Android. Si es du a terme aquest procés, l'aplicació no rep cap notificació sobre aquest canvi de situació de manera automàtica, i és per això que abans de realitzar les accions que requereixen un permís, s'ha de comprovar si aquest permís s'ha vist afectat cada vegada que es realitzi aquesta funcionalitat. Això s'aconsegueix mitjançant el següent codi:

```
if (ContextCompat.checkSelfPermission(MainActivity.this, Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED) {
   // Es te permis, realitzar accions desitjades
   } else {
   // No es te permis, demanar permis necessaris
}
```

Una vegada realitzades aquestes accions, es controla correctament la gestió de permisos d'Android, garantint els estàndards de protecció de dades que actualment està en vigor.

Per capturar una imatge amb la càmera o editar l'historial d'efectes s'ha d'utilitzar altres activitats. Per poder fer-ho s'ha de seguir unes pautes per a poder-ho implementar de forma correcta.

Desplaçament entre activitats

L'aplicació requereix diferents activitats, tant activitats creades per nosaltres com natives d'Android, com poden ser la càmera o la galeria d'imatges. Per a poder canviar d'activitat es requereix l'execució d'un codi específic. Per exemple, per obrir una activitat on aparegui la galeria d'imatges i poder escollir-ne una, el codi seria el següent:

```
Intent chooseImageIntent = new
Intent(Intent.ACTION_PICK,MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
startActivityForResult(chooseImageIntent, galleryRequestCode);
```

S'utilitza la classe Intent [24], que serveix per canviar d'activitat o realitzar certes accions. Aquest és un intent implícit, ja que no s'anomena directament quina és l'activitat a la qual es vol anar, sinó que es declara quina és l'acció que es vol realitzar i, una vegada s'executi l'intent, les aplicacions que puguin realitzar aquesta acció, podran ser cridades per dur-la a terme. Per exemple, en aquest cas, qualsevol aplicació que tingui la capacitat de fer que l'usuari esculli un element i en retorni certes dades, pot ser invocada.
En el primer paràmetre es transmet quines són les intencions que es pretén realitzar en l'altra activitat, i en el segon, quina és l'Uri en la qual s'ha de dur a terme aquesta acció.

Un altre cas més complex, és la interacció entre activitats creades expressament dins de l'aplicació, com la interacció entre l'activitat principal de l'app MainActivity i l'activitat per a l'edició de l'historial HistorialActivity. Per passar de la primera a la segona, el codi seria el següent:

```
Intent goHistorialActivityIntent = new Intent(MainActivity.this, HistorialActivity.class);
goHistorialActivityIntent.putExtra("KEY_URI", ima.getUri().toString());
goHistorialActivityIntent.putIntegerArrayListExtra("KEY_ID", getHistorialIds());
goHistorialActivityIntent.putIntegerArrayListExtra("KEY_VALUE", getHistorialValues());
startActivityForResult(goHistorialActivityIntent, goHistorialActivityRequestCode);
```

Aquest, en canvi, és un intent explícit, ja que sí que es nomena quina és l'activitat de la qual es vol marxar i quina és la que es vol llençar.
Observant els dos exemples, veiem que sempre hi ha dues accions que sempre s'han de dur a terme.

La primera és la creació d'un objecte de la classe Intent, on s'especifica què és el que es vol fer. En el cas de l'elecció d'una imatge de la galeria, podem observar que els paràmetres de creació de l'intent són 'Intent.ACTION_PICK'. Aquest paràmetre estableix que es vol seleccionar una dada dins d'un conjunt que estarà format pels elements que estiguin dins del directori de dades que es passa com a segon paràmetre 'MediaStore.Images.Media.EXTERNAL_CONTENT_URI'. En utilitzar aquest primer paràmetre, també es configura que el retorn de l'activitat de la qual es retornarà serà una Uri.

La segona és la invocació del mètode 'startActivityForResult()', la qual sempre rep com a paràmetres l'intent que s'ha creat anteriorment i un 'requestCode'. Aquest últim és un codi identificador que servirà per identificar de quina activitat es retorna. Això és necessari perquè per fer el viatge d'anada cap a una activitat es necessita unes línies de codi diferent per a cada activitat, però per fer el viatge de tornada, s'utilitza les mateixes línies de codi compartit, per tant s'ha de poder distingir de quina activitat es retorna en cada cas.

En el cas de passar de l'activitat MainActivity a l'activitat HistorialActivity, podem observar que hi ha unes quantes crides a mètodes que no són a l'altre exemple. Són els mètodes 'putExtra'. Aquests tenen la utilitat d'afegir les dades que es volen enviar cap a l'altra activitat. En el cas de l'elecció de la imatge de la galeria no és necessari, però en el cas de l'historial si, ja que precisament volem anar a l'altra activitat per a poder editar les dades que s'hi envia.
Hi ha diferents tipus de 'putExtra()' que permeten la transmissió de dades de les classes natives de Java (String, char, int, arrays de les classes anteriors...). La invocació d'aquest mètode es realitza passant com a primer paràmetre un identificador per a l'element que es transmet, que es passa com a segon paràmetre.

Una vegada es retorna de l'activitat a la qual s'ha anat, per controlar aquest retorn, se sobreescriu el mètode 'onActivityResult()' que proporciona la següent informació:

- int requestCode: quina activitat és de la que es retorna.
- int resultCode: el resultat de l'operació realitzada a l'altra activitat.
- Intent returnedIntent: dades que se'n retorna.

Anteriorment ja s'ha comentat la utilitat del 'requestCode'.

Pel que fa a 'resultCode' és una variable que indica quin ha estat el resultat de viatjar a l'altra activitat. En l'exemple de l'elecció de la imatge de la galeria, suposem que l'usuari escull una imatge entre totes les que pot triar. El 'resultCode' que es retornarà serà 'RESULT_OK' i indicarà que l'elecció s'ha dut a terme correctament. Suposem ara que l'usuari canvia d'opinió i ja no vol triar la imatge de la galeria, i la vol fer amb la càmera, per tant, decideix tirar endarrere. En aquest cas també es retorna de l'activitat de la galeria cap a la principal de la mateixa manera que en el cas que l'usuari hagués escollit la foto, però ara no hi ha cap dada de retorn de la galeria, ja que no s'ha escollit. Això podria comportar en un error en temps d'execució. La diferència de resultat es plasma en el 'resultCode' que en aquest cas retornaria 'RESULT_CANCELED'. Per poder evitar error d'execució i poder tenir un bon flux de programa s'ha de tenir això en compte.

El tercer paràmetre conté les dades retornades de l'altra activitat. Si en el cas anterior el mètode per afegir dades era el 'putExtra()' ara s'invocarà el mètode que proporciona la funció inversa. Aquest és 'getExtra()', en el qual es passarà com a paràmetre l'identificador que se l'hi ha passat anteriorment. Segueix les mateixes regles de funcionament que 'putExtra()'.

Animacions

Les animacions [25] aporten a l'aplicació un aspecte dinàmic i agradable de cara a la interacció amb l'usuari. Una app pot funcionar de forma completament correcta sense tenir cap mena d'animació però un dels objectius de qualsevol aplicació és que l'usuari gaudeixi utilitzant l'app, i una de les maneres d'aconseguir això és mitjançant les animacions.

En aquesta ocasió s'ha decidit animar l'aparició dels elements més significatius de l'activitat MainActivity cada vegada que aquests entrin en acció. Aquests són:

- TextView on es mostra el valor de la SeekBar.
- SeekBar.
- Botó d'acceptar.
- Botó de cancel·lar.
- Botó de l'efecte Equilibri de color.
- RecyclerView d'elecció d'efecte

En tots els casos s'ha decidit dur a terme el mateix tipus d'animació, així que el següent codi pot ser extrapolat en tots els casos, variant, lògicament, les referències als Views que s'apliquen:

```
myRecyclerView.setAlpha(0f);
myRecyclerView.setVisibility(View.VISIBLE);
myRecyclerView.animate()
        .setStartDelay(mShortAnimationDuration)
```

```
.alpha(1f)
.setDuration(mShortAnimationDuration)
.setListener(null);
```

Tots aquests mètodes són propis de la classe View, i per tant, es poden aplicar a qualsevol element de la pantalla que sigui subclasse de View.
Amb el primer mètode s'assigna quin és el nivell d'opacitat (transparència) des de la qual es vol iniciar l'animació. En aquest cas, el valor float 0 significa que és una opacitat del 0%, és a dir, totalment transparent. Seguidament s'assigna al View una visibilitat amb el valor View.VISIBLE, ja que volem que l'usuari pugui interactuar amb l'element un cop finalitzada l'animació.
Amb el mètode 'animate()' es procedeix a executar l'animació, i sobre aquest mateix mètode, que retorna un objecte de la classe ViewPropertyAnimator, s'executen un seguit d'altres mètodes:

- setStartDelay(): assigna un temps de retard en l'inici de l'animació.
- alpha(): estableix el valor d'opacitat del View en finalitzar l'animació.
- setDuration(): estableix el valor de duració de l'animació.
- setListener(): estableix el listener a l'animador de propietats.

En el nostre cas, la variable mShortAnimationDuration té assignat el valor que Android estima que ha de durar una animació curta. Per obtenir aquest valor es crida el següent mètode amb el següent paràmetre d'entrada:

```
getResources().getInteger(android.R.integer.config_shortAnimTime);
```

Amb el mètode 'alpha()' s'assigna l'opacitat final, que com que volem que sigui del 100%, s'ha de passar com a paràmetre el valor 1f. Per acabar, li assignem un listener null, ja que no necessitem saber quan s'acaba l'animació.

HistorialActivity

Aquesta activitat consisteix en una activitat exclusivament dedicada a l'edició de l'historial d'efectes. En ella hi ha els suficients elements perquè l'usuari pugui decidir si vol mantenir un efecte o esborrar-lo. Està formada únicament per una pantalla, la qual es crearà el corresponent arxiu xml per a la descripció del layout seguint el disseny anomenat "Historial". Està formada pels següents elements.

- Menú superior: per elegir les accions a realitzar, com per exemple guardar les modificacions realitzades a l'historial.
- ImageView: per poder visualitzar la imatge i els canvis que es realitzen a aquesta cada vegada que l'historial pateix una modificació.
- RecyclerView: per poder seleccionar quins efectes es volen desactivar i quins no.

Abans d'entrar en detall, es llistarà quin son els atributs i mètodes d'aquesta activitat:

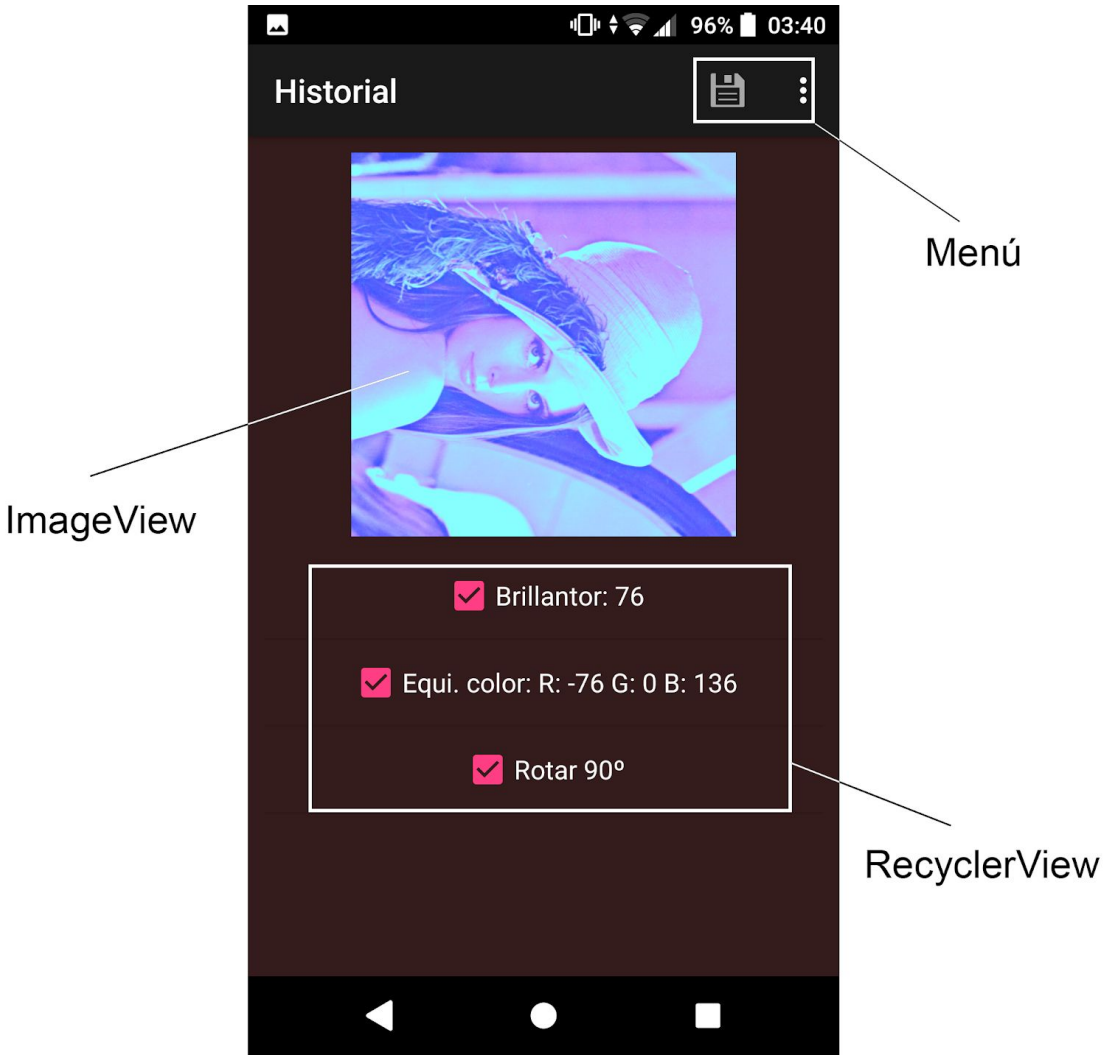
Atributs:

- public static String KEY_URI: variable per transmetre l'uri de la imatge entre activitats.
- public static String KEY_ID: variable per transmetre els id's dels efectes de l'historial entre activitats.
- public static String KEY_VALUE: variable per transmetre els valors dels efectes de l'historial entre activitats.
- private ImageView imageView: on es mostrarà la imatge.
- private RecyclerView.Adapter myAdapter: adaptador necessari per a treballar amb el RecyclerView.
- private Imatge ima: imatge que s'edita. Els detalls sobre la classe es troben en futurs apartats.
- private Uri uri: uri de la imatge que s'edita.
- private ArrayList<Effect> historial: historial d'efectes.
- private boolean[] effectStatus: estat d'aplicació dels efectes de l'historial.
- private boolean firstTime: variable necessària per a saber si es la primera vegada que s'executa una acció.

Mètodes:

- private void actualizeMiniature(): actualitza la imatge segons les dades de l'historial.
- public boolean historialHasChanged(): comprova si l'historial ha patit modificacions.
- public ArrayList<Integer> getHistorialIds(): obté una llista dels id's dels efectes de l'historial.
- public ArrayList<Integer> getHistorialValues(): obté una llista dels valors dels efectes de l'historial.

Per al correcte funcionament de cadascun d'aquests elements, s'ha assignat el següent comportament a cadascun d'ells:



Elements de l'activitat Historial.

Menú superior

Aquest menu realitza un total de 3 accions:

- Guardar els canvis efectuats.
- Seleccionar tots els efectes.
- Desseleccionar tots els efectes.

En la primera posició del menu es troba l'opció de guardar els canvis efectuats a l'historial. S'accepten els canvis realitzats en aquest sense realitzar comprovacions de modificació.

En la segona opció es permet a l'usuari la selecció de tots els efectes. Això permet estalviar temps si l'historial està compost per un llistat molt llarg d'efectes.

En la tercera i última posició es realitza l'opció complementaria a l'anterior: es desseleccionen tots els efectes a la vegada, també per estalviar temps.

Imageview

De la mateixa manera que en l'activitat principal, en aquest element es mostra la imatge amb els efectes de l'historial aplicats. Per defecte, mostra la imatge amb tots els efectes aplicats i quan es clica l'element, mostra la imatge original per a permetre fer comparacions. La imatge s'actualitza cada vegada que l'historial pateix una modificació, com es descriu al següent element.

RecyclerView

En aquest element es mostra el llistat d'efectes que s'han aplicat a la imatge, és a dir, els que hi ha emmagatzemats a l'historial. En arrossegar el dit de forma vertical es mostren les dades (nom i valor d'aplicació de l'efecte) de tots els efectes aplicats. Cada element d'aquest llistat està format per un CheckBox, un element natiu d'Android molt senzill en el que es mostra un text acompanyat d'una casella de verificació en la qual es mostrarà l'estat d'aplicació de l'efecte. Quan es clica aquest, es modifica l'estat d'aplicació de l'efecte. Si aquest estava aplicat, es desactiva, i si no estava aplicat, s'aplica. El resultat es veu plasmat en l'ImageView situat a la part superior.

Per aquest element s'ha creat un adaptador anomenat 'ListAdapter', seguint les indicacions d'apartats anteriors. Finalment, el RecyclerView està configurat per tal que el llistat es mostri en posició vertical i amb separador d'elements vertical, amb una mida fixa.

De la mateixa manera que en l'activitat anterior, s'ha sobreescrit el mètode 'onBackPressed()' perquè comprovi si s'ha realitzat alguna modificació en l'historial i es pregunti a l'usuari si aquesta ha de ser acceptada o descartada, en cas d'haver estat modificat.

4.2.2.2 Recursos

Per a cadascun dels diferents tipus de recursos descrits en apartats anteriors, en el nostre projecte s'ha creat o modificat els arxius següents:

anim:

- fade_in.xml: arxiu que conté l'animació d'entrada de l'activitat principal per a l'efecte splash screen.
- fade_out.xml: arxiu que conté l'animació de sortida de l'activitat 'SplashScreen' per a l'efecte splash screen.

drawable:

- background_splash.xml: es descriu quin és el contingut de la imatge que es mostrarà a l'activitat 'SplashScreen' i el color de fons d'aquesta.
- progressdialog.xml: arxiu de personalització dels quadres de diàleg.
- splash_photoditor.png: imatge que conté el logo de l'aplicació i que es mostra de fons en l'activitat 'SplashScreen'. S'assigna a aquesta en l'arxiu anteriorment esmentat.

layout:

- activity_historial.xml: arxiu en què es descriu el layout de l'activitat 'HistorialActivity'. En ell es descriu la posició de cadascun dels elements que la formen.
- activity_main.xml: de la mateixa manera que en l'arxiu anterior, aquí es descriu el layout de l'activitat 'MainActivity'
- effects_rv_item.xml: descripció del layout de cadascun dels elements que conformarà el RecyclerView d'elecció d'efecte de l'activitat principal.
- historial_rv_item.xml: layout de cada element que formarà el RecyclerView de l'activitat 'HistorialActivity' per a editar l'historial.

menu:

- historial_menu_layout.xml: elements dels quals es forma el menú de la part superior dreta de l'activitat 'HistorialActivity'.
- main_menu_layout.xml: elements dels quals es forma el menú de la part superior dreta de l'activitat 'MainActivity'.

mipmap:

- Arxius de tipus PNG creats amb el creador d'imatges Image Assets [26] que incorpora Android Studio i que optimitza la creació d'icones per al llançador de l'aplicació.

values:

- colors.xml: valors de codis de color que s'han d'externalitzar.
- strings: arxius que contenen les traduccions de les paraules utilitzades. Realment és una carpeta en la qual hi ha un arxiu anomenat strings.xml per a cada idioma en què es realitzi la traducció de cada paraula.
- styles.xml: arxiu que conté la personalització dels estils ja generats per Android dels elements que s'utilitza.

xml:

- file_paths.xml: en aquest fitxer es descriu la ruta on es guarden les fotos que es fan amb la càmera. Aquesta ruta és la que llegirà el FileProvider [27] per retornar la Uri que se sol·licita en realitzar una foto.

4.2.2.3 Classes

Durant el desenvolupament del projecte s'ha creat les diferents classes Java que s'han necessitat durant l'evolució del projecte per a representar els diferents tipus d'objectes amb els que treballa l'aplicació i les seves funcionalitats.

En aquest apartat trobareu una descripció completa sobre quin tipus d'objectes representa cada classe, per a què serveix, com està formada i perquè està implementada d'aquesta manera.

Classe Imatge

La peça clau d'aquesta aplicació són les imatges, i com a tal, s'ha decidit crear una classe pròpia per a poder tractar-les. S'anomena Imatge i segueix el següent funcionament.

El processament d'imatges pot ser una tasca que comporta un cert cost computacional normalment bastant elevat, sobretot si es treballa amb imatges de mides grans, és a dir, que contenen molts pixels que han de ser processats. Si a més a més, es contempla que molts usuaris d'avui en dia no disposen d'un smartphone amb processadors que tenen la suficient potència per realitzar tants càlculs i mostrar els resultats en temps real, això pot desenvolupar en què l'app es bloquegi. Per evitar aquest problema s'ha decidit optar per un sistema molt utilitzat en la producció de pel·lícules de cinema.

Com que, igual que en les imatges de mides grans, els vídeos en alta qualitat comporten molt temps de processament, el que s'han ingenyiat els editors de vídeo és un mètode bastant senzill i eficaç a l'hora de reduir aquest temps de processament.

Es converteix el vídeo original en un vídeo de baixa resolució, que té una mida considerablement molt més petita que el vídeo original. Per exemple, en un vídeo en format 4K (alta resolució) un sol frame té una mida de 3840x2160 pixels, és a dir, un total de 8.294.400 pixels, mentre que en format SVGA (baixa resolució) té una mida de 800x600, 480.000 pixels. La relació entre aquests dues mides és que el format SVGA conté 17,3 vegades menys pixels que el format 4K, que traduït en temps computacional és una diferència abismal.
S'ha de tenir en compte que això únicament és un sol frame, i quan es parla de vídeo es fa referència a un conjunt bastant gran de frames. Per a poder fer el símil amb les nostres necessitats, continuarem parlant en termes d'un sol frame.

Una vegada s'ha obtingut aquesta còpia en baixa qualitat del frame original, es procedeix a treballar sobre aquesta còpia, que, tot i no ser 100% fidel a la realitat, el resultat obtingut després d'aplicar-hi tot el processament és bastant semblant al que s'obté si aquest s'hagués aplicat sobre el frame original.

Quan ja s'han realitzat tots els canvis desitjats, es guarda en la memòria quines han estat totes les operacions que s'han aplicat sobre aquesta còpia, i finalment, s'apliquen en la imatge original, de manera que s'obté la imatge processada final.

Una de les característiques principals d'aquesta app és l'historial d'edició d'efectes, que és on es guarden tots els processos que s'apliquen a la imatge, així que es pot aprofitar encara més aquesta característica.

Una vegada explicada la idea bàsica del funcionament d'aquesta classe, analitzem quins són els elements que s'han necessitat per a l'elaboració d'aquesta classe.

Atributs estàtics (de classe):

- private static final int JPG: per definir que la imatge es guardarà en format JPG.
- private static final int PNG: per definir que la imatge es guardarà en format PNG.

Atributs no estàtics (d'objecte):

- private Bitmap bmpInicial: imatge en l'estat inicial, sense cap modificació.
- private Bitmap bmpEditat: imatge amb l'última modificació aplicada.
- private Bitmap bmpFinal: imatge que, quan s'està aplicant un efecte, mostra el resultat de l'aplicació d'aquest.
- private Uri uri: uri de l'arxiu imatge sobre la qual s'està treballant.
- private int limitAmple: ample límit que poden tenir els tres Bitmaps anteriors.
- private int limitAlt: altura límit que poden tenir els tres Bitmaps anterior.
- private int bmpAmple: ample dels tres Bitmap.
- private int bmpAlt: altura dels tres Bitmap.

Constructors:

- public Imatge(Context context, Uri uri, int limitWidth, int limitHeight): constructor que aplica un re escalament en la mida de la imatge. Aquest re escalament es du a terme en la crida interna a la funció 'setBitmap()' que aquest constructor realitza.
- private Imatge(Context context, Uri uri): constructor que no aplica re escalament i que únicament s'utilitza al final de tot el procés quan es vol aplicar els efectes a la imatge original i escriure el fitxer.

Mètodes:

- public Uri getUri(): retorna uri.
- public Bitmap getBitmapInicial(): retorna bmpInicial.
- public Bitmap getBitmapEditat(): retorna bmpEditat.
- public Bitmap getBitmapFinal(): retorna bmpFinal.
- public void resetBitmaps(): assigna a bmpEditat i a bmpFinal el bmpInicial.
- private void setBitmap(Bitmap bitmapIn): assigna el Bitmap d'entrada als tres atributs de tipus Bitmap restringint la mida al Bitmap que retorna la funció 'resizeToLimits(Bitmap bitmap)'.
- private Bitmap resizeToLimits(Bitmap bitmap): retorna un Bitmap canviat de mida tenint en compte els valors dels atributs 'limitAmple' i 'limitAlt', utilitzant la funció 'resizeBitmapTo(Bitmap bm, int newWidth, int newHeight)' i mantenint la relació d'aspecte original de la imatge.
- private Bitmap resizeBitmapTo(Bitmap bm, int newWidth, int newHeight): retorna un Bitmap amb la mida passada en els paràmetres d'entrada.
- public void applyEffect(Effect e): aplica l'efecte d'entrada a bmpEditat invocant el mètode 'execute()' de l'Effect que es passa com a paràmetre i es guarda el resultat a bmpFinal.
- public void applyHistorial(ArrayList<Effect> historial): aplica un conjunt d'efectes emmagatzemats a 'historial' utilitzant la funció 'applyEffect(Effect e)' i es crida el mètode 'substitueixEditat()'.
- public void substitueixEditat(): substitueix bmpEditat per bmpFinal.
- public void escriuJPEG(MainActivity activity, ArrayList<Effect> historial, int quality): crea un objecte de la classe interna 'SaveImage' per tal de guardar la imatge en format JPG amb la qualitat 'quality'.
- public void escriuPNG(MainActivity activity, ArrayList<Effect> historial, int quality): crea un objecte de la classe interna 'SaveImage' per tal de guardar la imatge en format PNG amb la qualitat 'quality'.
- private String generePhotoName(): genera un nom per a l'arxiu que es guardarà tenint en compte la data i l'hora en què es realitza la petició, en format '2017.12.15_20.31.46' S'utilitza la classe Calendar [28] per a l'obtenció d'aquestes dades.

Classe interna:
A causa de la complexitat d'aquesta classe s'explicarà de manera independent.

- private class SaveImage extends AsyncTask<Void, Integer, Boolean>

El cicle de vida d'un objecte de la classe Imatge és el següent:
Es crea un objecte utilitzant el constructor 'Imatge(Context context, Uri uri, int limitWidth, int limitHeight)', que recordem que realitza un re escalament en les mides de la imatge, en aquest cas fa que la mida dels Bitmaps que conformen la imatge no sigui superior a la mida de l'ImageView que mostra aquesta imatge.
En aquest pas és on es veu reflectit el mètode per estalviar temps i cost computacional.

Tot seguit l'usuari aplica un efecte a l'objecte utilitzant el mètode 'applyEffect(Effect e)'. Si l'usuari accepta el resultat d'aquest efecte, es crida el mètode 'substitueixEditat()' de

manera que el Bitmap on es guardava la imatge sobre la qual s'aplica l'efecte queda substituït pel resultat d'aquest. Si decideix cancel·lar-lo, simplement no s'aplica aquest últim mètode.
Cada vegada que s'accepta un efecte, aquest s'afegeix a l'historial, de manera que queden emmagatzemats per a la seva posterior aplicació en la imatge final.
De la mateixa manera, cada vegada que s'accepta o es cancel·la un efecte, s'actualitza el contingut de l'ImageView mostrant-hi el Bitmap corresponent. En cas d'acceptar l'efecte, el que retorna el mètode 'getBitmapFinal()', i en cas de cancel·lar-lo, el que retorna el mètode 'getBitmapEditat()'.
Quan l'usuari decideix que el resultat mostrat en pantalla és el resultat final, procedeix a guardar la imatge. Es crida la funció 'escriuJPEG(MainActivity activity, ArrayList<Effect> historial, int quality)' o 'escriuPNG(MainActivity activity, ArrayList<Effect> historial, int quality)' segons elecció de l'usuari, que crea un objecte de la classe SaveImage, la qual s'explica seguidament i que està exclusivament dedicada a aplicar els efectes de l'historial a la imatge original i escriure el fitxer resultant.

SaveImage

La creació d'aquesta classe ha estat necessària perquè el procés d'escriptura del fitxer resultant és una tasca que comporta cert temps de computació. Això desencadena en un bloqueig momentani de la pantalla de l'usuari que depèn de la capacitat de processament del dispositiu. Si aquest temps es prolonga massa, pot resultar en el fet que l'usuari cregui que l'aplicació està fallant i tancar-la. És per això que les tasques que requereixen cert temps de processament s'executen en 'background', és a dir, en segon pla. D'aquesta manera es pot continuar interactuant amb l'aplicació mentre altres tasques es duen a terme.

Com s'ha comentat, el processament d'imatges grans i l'escriptura del corresponent arxiu requereix cert temps de computació, així que es decideix crear una classe interna dins de la classe Imatge que extengui la classe AsyncTask<> [29]. Aquesta és una classe abstracta que permet personalitzar quina és la tasca que es vol implementar en segon pla.

A l'hora d'extendre aquesta classe s'ha d'implementar el mètode 'doInBackground()' de forma obligatòria, però a causa de les necessitats de l'aplicació s'ha optat per implementar-ne els quatre mètodes que es poden sobreescriure. Aquests són els següents:

- onPreExecute
- doInBackground
- onPostExecute
- onProgressUpdate

En el nostre cas es vol fer les següents tasques:

- Aplicar tots els efectes de l'historial a la imatge original.
- Escriure el fitxer resultant.

A més a més, durant el procés, es vol mostrar un quadre de diàleg que informi l'usuari quina operació s'està realitzant en cada moment i quin és el percentatge d'operacions realitzades.

Una vegada sabent aquesta informació, s'opta per extendre la classe 'AsyncTask' de la següent manera i amb els següents paràmetres:
AsyncTask<Classe 1, Classe 2, Classe 3>:

- Classe 1: classe de l'objecte que es passa al mètode 'doInBackground()', en aquesta ocasió, Void.
- Classe 2: classe de l'objecte que es passa a 'onProgressUpdate' i que s'envia a través del mètode 'publishProgress()' dins del mètode 'doInBackground()', en aquesta ocasió, Integer.
- Classe 3: classe de l'objecte que retorna el mètode 'doInBackground()' i que es passa al mètode 'onPostExecute()', en aquesta ocasió, Boolean.

A part d'això, la classe també compta amb atributs propis i alguns mètodes. Aquests són els descrits a continuació:

Atributs:

- private ProgressDialog progressDialog: quadre de diàleg que mostrarà la informació.
- private MainActivity activity: activitat sobre la qual es mostrarà el quadre de diàleg.
- private Uri uri: uri de la imatge original sobre la qual es treballa.
- private int format: format de la imatge a guardar.
- private ArrayList<Effect> historial: conjunt d'efectes que s'aplicaran a la imatge.
- private Uri uriFinal: uri del fitxer que s'ha escrit com a imatge final.
- private int actualFxIndex: índex de l'efecte que està aplicant-se actualment.
- private String directori_foto: directori on s'escriurà el fitxer.
- private int quality: qualitat de compressió.

Constructor:

- private SaveImage(MainActivity activity, Uri uri, ArrayList<Effect> historial, int format, int quality): s'assigna l'activitat en la qual es mostrarà el quadre de diàleg, la uri de la imatge original que s'aplicaran els efectes, l'historial amb els efectes que s'aplicaran a la imatge, el format de l'arxiu a guardar i la qualitat de compressió d'aquest. També s'executa el mètode 'customizeProgressDialog()'.

Mètodes:

- private void customizeProgressDialog(): personalització de les propietats del ProgressDialog () amb els paràmetres: títol, missatge inicial, cancelabilitat del quadre de diàleg i estil.

En cadascun dels mètodes a implementar es duen a terme les següents accions:

- onPreExecute: es mostra el quadre de diàleg en el qual s'indica quina operació s'està realitzant i el percentatge actual de progrés, aplicant el mètode 'show()' sobre l'objecte progressDialog.

- doInBackground: es crea un objecte Imatge aplicant el constructor que no aplica un re escalament a la imatge, aplica tots els efectes que es troben emmagatzemats a l'historial d'efectes, genera un nom per al fitxer i crea el directori on s'escriurà el fitxer si no existeix. Finalment s'escriu el fitxer, i mentre realitza cada acció, es va actualitzant el percentatge d'operacions realitzades per mostrar al quadre de diàleg.

- onPostExecute: s'oculta el quadre de diàleg amb el mètode 'dismiss()', i si l'operació ha anat correctament, s'afegeix la nova imatge a la galeria mitjançant un objecte de la classe Intent en el qual s'estableix l'acció a realitzar com a Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, s'afegeix com a dada a aquest Intent l'Uri del fitxer que s'ha escrit amb el mètode 'setData()' i finalment s'invoca el mètode 'sendBroadcast()' passant com a paràmetre aquest Intent, sobre l'atribut activity. Finalment es mostra un Toast informant de si la tasca s'ha realitzat correctament o no.

- onProgressUpdate: modifica el percentatge de progrés actual i el missatge a mostrar en el quadre de diàleg segons si s'estan aplicant els efectes, escrivint el fitxer o si s'ha acabat el procés.

Classe Effect

La classe Effect vol representar els diferents tipus de processaments que es poden aplicar a una imatge, tals com canviar la brillantor, el contrast, la saturació, etc. És una classe abstracta, és a dir, no es pot crear un objecte Effect com a tal, sinó que està pensada per a què es crei un Effect d'una de les subclasses que es veurà més endavant i que la classe Effect n'és la superclasse.

Això neix de la necessitat d'implementar un historial d'edició, un llistat on s'emmagatzemen els efectes que s'han aplicat sobre la imatge amb la finalitat de poder afegir-hi efectes o esborrar-los, però sobretot per poder dur a terme el mètode per a optimitzar el cost de computació comentat en l'apartat de la classe Imatge.

Els atributs i mètodes que s'ha necessitat per a la classe Effect són:

Atributs estàtics:

- public static final int MAX_BAR_VALUE: valor màxim de la SeekBar.
- public static final int TOTAL_NUM_EFFECTS: nombre total d'efectes.
- public static final int INDEX_BRILLO: índex per a l'efecte 'Brillo'.
- public static final int INDEX_CONTRAST: índex per a l'efecte 'Contrast'.
- public static final int INDEX_SATURACIO: índex per a l'efecte 'Saturacio'.
- public static final int INDEX_EQUILIBRICOLOR: índex per a l'efecte 'EquilibriColor'.
- public static final int INDEX_BALANCBLANC: índex per a l'efecte 'BalancBlanc'.
- public static final int INDEX_DESENFOCAR: índex per a l'efecte 'Desenfocar'.
- public static final int INDEX_RETALLAR: índex per a l'efecte 'Retallar'.
- public static final int INDEX_ROTACIO90: índex per a l'efecte 'Rotacio90'.
- public static final int INDEX_ROTACIOVERTICAL: índex per a l'efecte 'RotacioVertical'.
- public static final int INDEX_ROTACIOHORITZONTAL: índex per a l'efecte 'RotacioHoritzontal'.
- public static final int INDEX_ROTACIOANGLE: índex per a l'efecte 'RotacioAngle'.
- public static final int INDEX_BLANCNEGRE: índex per a l'efecte 'BlancNegre'.
- public static final int INDEX_INVERTIRCOLOR: índex per a l'efecte 'InvertirColors'.
- public static final int INDEX_LLUMS: índex per a l'efecte 'Llums'.
- public static final int INDEX_OMBRES: índex per a l'efecte 'Ombres'.
- public static final int INDEX_NIVELLBLANC: índex per a l'efecte 'NivellBlanc'.
- public static final int INDEX_NIVELLNEGRE: índex per a l'efecte 'NivellNegre'.
- public static final int INDEX_OPACITAT: índex per a l'efecte 'Opacitat'.
- public static float margeClickAreaRetallar: pixels de marge per decidir si s'ha clicat dins del requadre de l'efecte 'Retallar' o fora.
- public static float distMinRetallar: distància mínima que hi pot haver entre dos costats paral·lels del requadre de l'efecte 'Retallar'

Atributs no estàtics:

- protected Context context: necessari per obtenir el nom de l'efecte dels recursos.
- protected String name: nom de l'efecte.
- protected int id: índex de l'efecte.
- private int barValue: valor d'aplicació de l'efecte.
- private int defaultValue: valor per defecte de l'efecte.
- protected int GUI: tipus de layout que requereix l'efecte

Constructor:

- Effect(Context context): context de l'activitat en què es crea.

Mètodes:

- public String getName(): retorna el nom de l'efecte.
- public Integer getId(): retorna l'índex de l'efecte.
- public int getGUI(): retorna quin layout necessita l'efecte.
- public int getBarValue(): retorna el valor d'aplicació de l'efecte.
- public int getDefaultValue(): retorna el valor per defecte de l'efecte.
- public void setBarValue(int barValue): assigna el valor d'aplicació de l'efecte.
- protected void setDefaultValue(int defaultValue): assigna el valor per defecte a l'efecte.
- abstract public Bitmap execute(Bitmap bmp): mètode abstracte per aplicar un efecte.
- abstract public String toString(): mètode abstracte per obtenir el nom i el valor d'aplicació de l'efecte.
- public static Effect fromIndex(Context context, int codiFx): crea un objecte d'alguna de les subclasses d'Effect segons el codiFx donat. En cas que codiFx no es correspongui amb cap índex dels anteriorment nomenats, es retorna 'null'.

Per a poder emmagatzemar en un llistat tots els efectes que s'han d'aplicar en una imatge, es necessita encapsular tant quin és l'efecte que s'aplica com el seu valor en un objecte. Aquesta és la missió principal de la classe Effect. A partir d'aquí, es poden emmagatzemar en un ArrayList<Effect>.

El següent problema que se'ns planteja és com aplicar un efecte, que conté les dades emmagatzemades en els seus atributs i el codi a executar en els seus mètodes, a la nostra classe Imatge. La solució adequada és utilitzar un patró de disseny anomenat 'Command Pattern', patró de comandes, en català.

Command Pattern

El Command Pattern [30] és un patró de disseny que permet encapsular ordres en objectes per tal de poder ser executades per un receptor. Aquest fet ens resulta extremadament útil, ja que en convertir cadascuna de les ordres que volem executar en un objecte de la classe Effect ens permetrà dues coses:

- Emmagatzemar en un llistat tots els efectes que s'han d'aplicar a l'objecte Imatge i tenir la capacitat d'afegir-ne o esborrar-ne.
- Crear subclasses que extenguin la classe Effect i així facilitar la implementació de nous efectes.

En aquest patró de disseny bàsicament s'hi pot trobar 3 elements clau:

- Client: és el que crea una comanda, que va destinada a un receptor. En el nostre cas, els botons que creen cadascun dels efectes.
- Receptor: és el destinatari de la comanda i el que realitzarà les accions que la comanda dicta. En el nostre cas, l'objecte de la classe Imatge.
- Comanda: és l'objecte en el qual queden encapsulades les dades que s'han de transmetre. En el nostre cas, la classe Effect.

El funcionament del patró és el següent. L'usuari fa clic en algun dels botons que inicien les accions necessàries per aplicar un efecte. Això es reflecteix en la creació d'un objecte de la classe Effect, tot i que realment serà un objecte d'una de les subclasses d'aquesta (Brillo, Contrast...). S'entrarà en detalls sobre aquest tema en punts següents.
Quan això passa, aquest objecte llegeix els paràmetres sobre quin tipus d'efecte és (índex) i quin és el seu valor d'aplicació (barValue).
Seguidament, s'envia la comanda al receptor, en aquest cas l'objecte Imatge. Això es fa gràcies al mètode que té implementat la classe Imatge, 'applyEffect(Effect e)'.
Si s'entra en la implementació d'aquest mètode, es pot observar que el que es realitza realment aquí és la crida al mètode abstracte 'execute()'. En ser aquest un mètode abstracte de la classe Effect, significa que no s'implementarà mai directament en aquesta classe sinó que es farà en cadascuna de les subclasses, i gràcies a aquest fet, es pot descriure el comportament que té cada efecte per separat. És a dir, és en la implementació a nivell de subclasse del mètode 'execute()' on realment hi ha la diferència de codi que executarà cada efecte, ja que per modificar el contrast d'una imatge no s'ha de realitzar les mateixes operacions que per convertir-la en blanc i negre, per exemple.
En la implementació del mètode 'execute()' es troba el codi dedicat a cada efecte, però tots retornen el mateix tipus de dades, que és el Bitmap resultant d'aplicar-hi el processament determinat, és a dir, la comanda una vegada ha estat finalitzada.

SubClasses d'Effect

Com ja s'ha comentat, es crearan diferents subclasses que extendran la classe Effect per tal de poder implementar cadascun dels diferents efectes que es vol desenvolupar.

El llistat final d'efectes desenvolupats es troba a continuació:

- Balanç de blanc
- Blanc i negre
- Brillantor
- Contrast
- Desenfocar
- Equilibri de color
- Invertir colors
- Llums
- Nivell de Blanc
- Nivell de negre
- Ombres
- Opacitat
- Retallar
- Rotació 90°
- Rotació d'angle
- Rotació horitzontal
- Rotació vertical
- Saturació

Per a la creació de cadascun d'aquests efectes es crearà una subclasse que extendrà la classe Effect, i com a tal, haurà d'implementar els mètodes que aquesta té definits com a abstractes i un constructor.

El constructor de tots els mètodes tenen una implementació bastant semblant a la descrita a continuació. S'agafa com a exemple el constructor de la classe Brillo:

```
Brillo(Context context) {
        super(context);
        this.setDefaultValue(Effect.MAX_BAR_VALUE / 2);
        this.setBarValue(getDefaultValue());
        this.id = Effect.INDEX_BRILLO;
        this.name = this.context.getString(R.string.brightness);
        this.GUI = GUIValue.SEEKBAR_SIMPLE;
}
```

En primer lloc es crida el constructor de la classe Effect, que assigna el context donat a l'atribut context de la classe Effect. Això és necessari per a poder agafar, com es veurà a continuació, el String que conté el nom de l'efecte dels recursos interns d'Android. Seguidament es procedeix amb l'assignació dels valors que caracteritzen cada efecte:

- defaultValue: s'estableix quin és el valor per defecte de l'efecte. Aquest valor és el que, quan s'aplica l'efecte amb aquest valor, el resultat és el mateix que si aquest no fos aplicat.
- barValue: s'assigna al valor d'aplicació de l'efecte el valor per defecte d'aquest.
- id: s'assigna un índex a l'efecte. Aquest ha de ser un dels valors estàtics de la classe Effect.
- name: nom de l'efecte. És el que es mostrarà en el RecyclerView on s'escull quin efecte s'aplicarà. S'obté invocant el mètode 'getString()' i passant com a paràmetre el nom del recurs desitjat. Es fa d'aquesta manera per tal de disposar de traduccions dels noms de cada efecte.
- GUI: s'estableix quin tipus de layout requereix l'efecte per a poder configurar-lo. Això es decideix segons quin tipus d'efecte és: simple, complex, o on/off. Segons aquest tipus es mostrarà una SeekBar, una SeekBar i un Button o s'ocultarà la SeekBar. Ha de ser un dels valors estàtics de la classe GUIValue.

El següent mètode que s'implementa és 'toString()'. Aquest mètode té la finalitat de convertir l'objecte en un String que mostri la informació sobre els atributs bàsics d'aquest perquè l'usuari tingui el coneixement sobre les dades bàsiques d'aquest. En el nostre exemple:

```
@Override
public String toString() {
        return name + ": " + (getBarValue() - (Effect.MAX_BAR_VALUE / 2)) * 2;
}
```

S'observa que en el cas de Brillo es retorna el nom de l'efecte i el valor d'aplicació d'aquest aplicant-hi un factor de correcció. Això és degut al fet que el valor real que aplica l'efecte no és en tots els casos el valor que retorna la SeekBar. S'ha de tenir en compte que la SeekBar està configurada de tal manera que retorna un valor dins del rang 0 i Effect.MAX_BAR_VALUE, que es troba assignat a un valor de 200. Alguns efectes poden treballar directament amb valors dins d'aquest rang, però en alguns casos, com per exemple en l'efecte 'Desenfocar' es treballa amb un rang més petit i utilitzant decimals. És per això que dins del mètode 'execute()' de cada efecte es realitza una sèrie d'operacions per a realitzar una conversió d'aquestes dades.

L'últim mètode que s'implementa és precisament aquest, el mètode 'execute()'. És el que conté el codi que s'executa per tal de processar la imatge. En cada efecte hi ha diferents conceptes per tal d'aconseguir modificar la qualitat desitjada de la imatge. Com a entrada, rep el Bitmap que es vol modificar, i en la sortida es retorna el Bitmap resultat. En aquest apartat s'ha de tenir especial cura en el tipus d'operacions que es realitza, ja que en algunes es mescla diferents tipus de variables, com per exemple float i int, i poden desencadenar en

resultants no correctes. Tot i que en les descripcions de les implementacions de cada efecte que s'ha explicat en l'apartat 2. Background això no es contempla, sí que s'ha tingut en compte a l'hora d'implementar el codi en l'aplicació utilitzant la funció cast quan ha estat convenient.

Classe BalancBlanc

Característiques:

- Valor per defecte: Effect.MAX_BAR_VALUE / 2
- Rang de valors d'aplicació: de -50 a 50
- Índex: Effect.INDEX_BALANCBLANC
- Nom del recurs: R.string.white_balance
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Per qüestions d'implementació s'utilitzarà el mètode 2 explicat a l'apartat 2. Background, ja que requereix un menor nombre d'operacions de càlcul.

Classe BlancNegre

Característiques:

- Valor per defecte: 0
- Rang de valors d'aplicació: no correspon
- Índex: Effect.INDEX_BLANCNEGRE
- Nom del recurs: R.string.black_white
- Tipus d'efecte i GUI: GUIValue.ON_OFF

Classe Brillo

Característiques:

- Valor per defecte: Effect.MAX_BAR_VALUE / 2
- Rang de valors d'aplicació: de -200 a 200
- Índex: Effect.INDEX_BRILLO
- Nom del recurs: R.string.brightness
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Classe Contrast

Característiques:

- Valor per defecte: Effect.MAX_BAR_VALUE / 2
- Rang de valors d'aplicació: de 0.22 a 1.76
- Índex: Effect.INDEX_CONTRAST
- Nom del recurs: R.string.contrast
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Classe Desenfocar

Característiques:

- Valor per defecte: 0
- Rang de valors d'aplicació: de 0.0 a 25.0
- Índex: Effect.INDEX_DESENFOCAR
- Nom del recurs: R.string.defocus
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Per a implementar aquest efecte s'ha utilitzat la classe RenderScript [31], que proporciona un seguit d'efectes predefinits, i que utilitzarem, a causa de la dificultat en la implementació del codi necessari per dur a terme aquest efecte de forma manual. El codi utilitzat és el següent:

```
RenderScript rs = RenderScript.create(this.context);
ScriptIntrinsicBlur blur = ScriptIntrinsicBlur.create(rs, Element.U8_4(rs));
Allocation tmpIn = Allocation.createFromBitmap(rs, bmp);
Allocation tmpOut = Allocation.createFromBitmap(rs, outputBitmap);
 if (valor > 0f) {
        blur.setInput(tmpIn);
        blur.setRadius(valor);
        blur.forEach(tmpOut);
        tmpOut.copyTo(outputBitmap);
        rs.destroy();
}
```

Es crea un objecte de la classe ScriptIntrinsicBlur [32] amb el mètode 'create()' passant com a paràmetres un objecte RenderScript i un Element.U8_4() que rep també com a paràmetre aquest mateix objecte RenderScript. Seguidament es creen dos objectes Allocation [33], que serà on s'aplica el desenfoc. Es configurarà l'objecte blur amb els mètodes 'setInput()', 'setRadius()' i 'forEach()' tenint en compte que el valor del radi ha de ser superior a 0. Finalment es transforma l'objecte Allocation en el Bitmap que es retorna i es destrueix el RenderScript per alliberar memòria.

Classe EquilibriColor

Característiques:

- Valor per defecte: Effect.MAX_BAR_VALUE / 2
- Rang de valors d'aplicació: de -200 a 200
- Índex: Effect.INDEX_EQUILIBRICOLOR
- Nom del recurs: R.string.color_balance
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_COMPLEX

A causa de la idea inicial de la utilització de tres Seekbar per a implementar aquest efecte i als canvis de disseny efectuats, s'ha implementat una classe especial per a poder adaptar aquest efecte a les necessitats especials que aquest requereix. Això s'explica dins l'apartat 4.3.1.5 Classe TransmitValues.

Classe InvertirColors

Característiques:

- Valor per defecte: 0
- Rang de valors d'aplicació: no correspon
- Índex: Effect.INDEX_INVERTIRCOLOR
- Nom del recurs: R.string.invert_color
- Tipus d'efecte i GUI: GUIValue.ON_OFF

Classe Llums

Característiques:

- Valor per defecte: Effect.MAX_BAR_VALUE
- Rang de valors d'aplicació: de 0.0 a 1.0
- Índex: Effect.INDEX_LLUMS
- Nom del recurs: R.string.lights
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Aquest efecte disposa de dues maneres diferents de ser aplicat, tal com s'ha explicat en l'apartat 2. Background, però com que la reconstrucció del senyal de luminància necessita un nombre major d'operacions, s'implementarà aplicant aquesta fórmula sobre els valors dels canals RGB.

Classe NivellBlanc

Característiques:

- Valor per defecte: Effect.MAX_BAR_VALUE
- Rang de valors d'aplicació: de 0 a 255
- Índex: Effect.INDEX_NIVELLBLANC
- Nom del recurs: R.string.white_level
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Com s'ha fet referència en l'apartat 2. Background, aquest efecte es pot aplicar tant en les components RGB com en la luminància, però com que la reconstrucció de la luminància requereix un nombre major d'operacions, s'implementarà aplicant aquesta limitació sobre els valors dels canals RGB.

Classe NivellNegre

Característiques:

- Valor per defecte: 0
- Rang de valors d'aplicació: de 0 a 255
- Índex: Effect.INDEX_NIVELLNEGRE
- Nom del recurs: R.string.black_level
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Se segueix el mateix criteri que en el cas de la classe NivellBlanc, ja que ambdós tenen els mateixos requeriments i problemes.

Classe Ombres

Característiques:

- Valor per defecte: 0
- Rang de valors d'aplicació: de 1.0 a 0.0
- Índex: Effect.INDEX_OMBRES
- Nom del recurs: R.string.shadows
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Semblant a la classe Llums, aquest efecte també disposa de dues maneres diferents de ser aplicat, i pels mateixos motius que en l'efecte anterior, s'opta per aplicar aquesta fórmula sobre els valors dels canals RGB.

Classe Opacitat

Característiques:

- Valor per defecte: Effect.MAX_BAR_VALUE
- Rang de valors d'aplicació: de 0 a 255
- Índex: Effect.INDEX_OPACITAT
- Nom del recurs: R.string.opacity
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Classe Retallar

Característiques:

- Valor per defecte: depèn de cada imatge
- Rang de valors d'aplicació: no correspon
- Índex: Effect.INDEX_RETALLAR
- Nom del recurs: R.string.crop
- Tipus d'efecte i GUI: GUIValue.ON_OFF

A causa de les necessitats especials d'aquest efecte, el valor d'aplicació es rep d'una manera diferent de la resta. Això s'explica en l'apartat 4.3.1.5, on s'explica el funcionament de la classe TransmitValues, en el subapartat 'Retallar'.
Una vegada s'ha llegit els valors de la classe TransmitValues, tan sols s'ha de reconvertir aquests en valors aplicant les operacions inverses al procés de codificació.

Classe Rotacio90

Característiques:

- Valor per defecte: 0
- Rang de valors d'aplicació: no correspon
- Índex: Effect.INDEX_ROTACIO90
- Nom del recurs: R.string.rotation90
- Tipus d'efecte i GUI: GUIValue.ON_OFF

S'utilitza un objecte de la classe Matrix [34] per tal de girar el Bitmap. Es crida el mètode 'setRotate()' passant com a paràmetre el valor 90, que són els graus que es vol girar la imatge.

Classe RotacioAngle

Característiques:

- Valor per defecte: Effect.MAX_BAR_VALUE / 2
- Rang de valors d'aplicació: de -180 a 180
- Índex: Effect.INDEX_ROTACIOANGLE
- Nom del recurs: R.string.rotation
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

S'utilitza el mateix codi que en la classe Rotacio90, però en aquesta ocasió es crida el mètode 'setRotate()' passant com a paràmetre el valor en graus que es vol girar la imatge.

Classe RotacioHoritzontal

Característiques:

- Valor per defecte: 0
- Rang de valors d'aplicació: no correspon
- Índex: Effect.INDEX_ROTACIOHORITZONTAL
- Nom del recurs: R.string.horitzontal_rotation
- Tipus d'efecte i GUI: GUIValue.ON_OFF

S'utilitza un objecte de la classe Matrix per tal de girar el Bitmap. Es crida el mètode 'postScale()' passant com a paràmetres la inversió de l'eix X (-1), la no inversió de l'eix Y (1) i el centre de la imatge en l'eix X (ample / 2) i el centre de la imatge en l'eix Y (alt / 2), que són les coordenades sobre les quals es basa aquesta inversió.

Classe RotacioVertical

Característiques:

- Valor per defecte: 0
- Rang de valors d'aplicació: no correspon
- Índex: Effect.INDEX_ROTACIOVERTICAL
- Nom del recurs: R.string.vertical_rotation
- Tipus d'efecte i GUI: GUIValue.ON_OFF

Igual que en la classe RotacioHoritzontal, es crida el mètode 'postScale()' passant com a paràmetres la no inversió de l'eix X (1), la inversió de l'eix Y (-1) i el centre de la imatge en l'eix X (ample / 2) i el centre de la imatge en l'eix Y (alt / 2), que són les coordenades sobre les quals es basa aquesta inversió.

Classe Saturacio

Característiques:

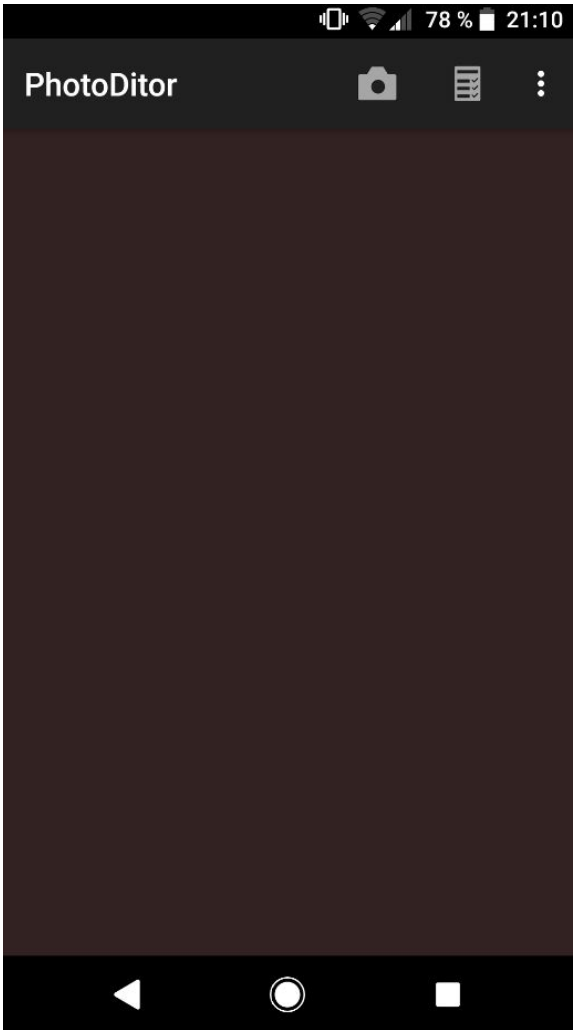
- Valor per defecte: Effect.MAX_BAR_VALUE / 2
- Rang de valors d'aplicació: de 0.1 a 7.4
- Índex: Effect.INDEX_SATURACIO
- Nom del recurs: R.string.saturation
- Tipus d'efecte i GUI: GUIValue.SEEKBAR_SIMPLE

Classe GUIValue

Aquesta classe té com a únic objectiu facilitar l'obtenció del valor del tipus de layout que es necessita en cada ocasió. Bàsicament està formada per 5 atributs estàtics que permeten associar a cada layout un valor, que mitjançant la funció 'setGUI()' de l'activitat MainActivity, es decideix quins elements han d'estar a la vista i quins no. Els atributs són els següents:

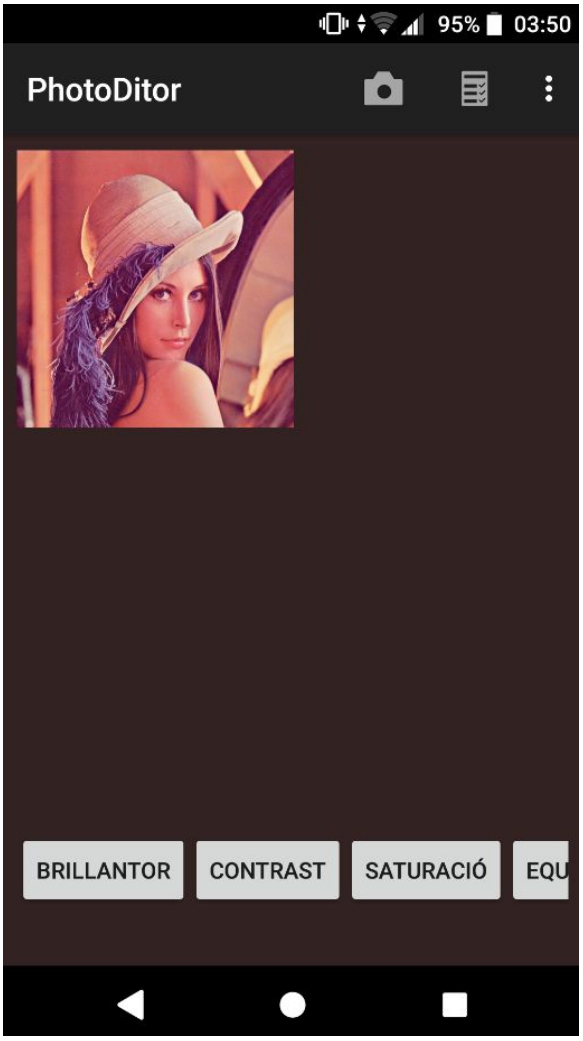
- public static final int HIDE_ALL: valor per ocultar tots els elements.
- public static final int ONLY_RECYCLERVIEW: valor per a selecció d'efecte.
- public static final int SEEKBAR_SIMPLE: valor per a efectes de tipus simple.
- public static final int SEEKBAR_COMPLEX: valor per a efectes de tipus complex.
- public static final int ON_OFF: valor per a efectes de tipus on/off.

El primer layout que es visualitza quan s'entra a l'activitat MainActivity és el GUIValue.HIDE_ALL, en el que únicament es mostra els elements del menú superior, exceptuant la barra de navegació (barra inferior), que es mostra sempre.



Captura de pantalla amb el layout inicial.

Una vegada s'ha seleccionat una imatge, es mostra aquesta dins de l'ImageView d'aquesta l'activitat i es procedeix a canviar de layout, mostrant en aquesta ocasió el layout GUIValue.ONLY_RECYCLERVIEW, que activa el RecyclerView perquè l'usuari seleccioni l'efecte a aplicar a la imatge.

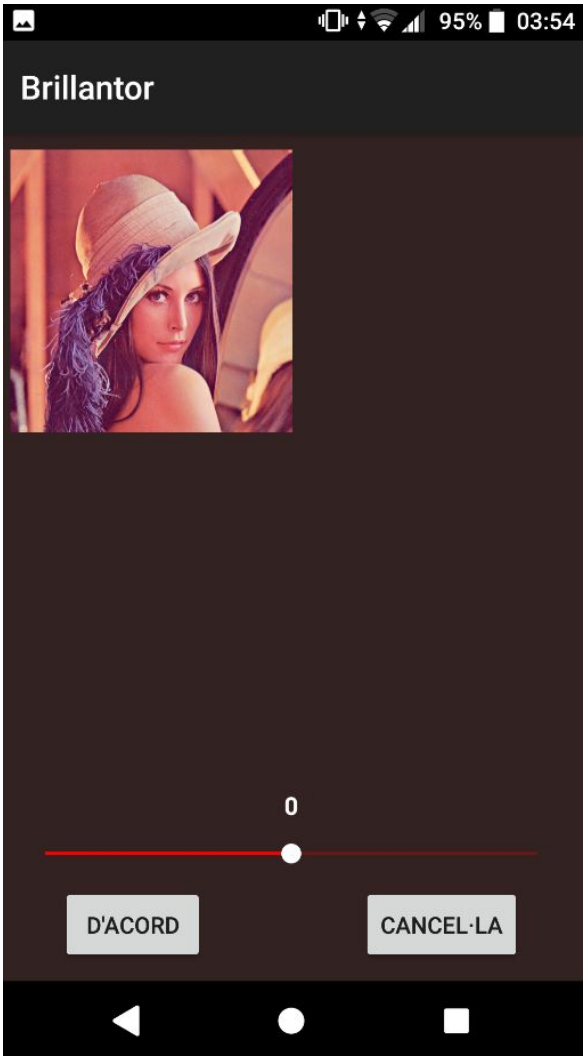


Captura de pantalla amb el layout de selecció d'efecte.

Segons el tipus d'efecte elegit, es mostra el tipus de layout necessari per a poder interactuar correctament amb aquest efecte:

Efectes simples (es controlen amb una sola SeekBar): tenen assignat el layout amb el valor GUIValue.SEEKBAR_SIMPLE:

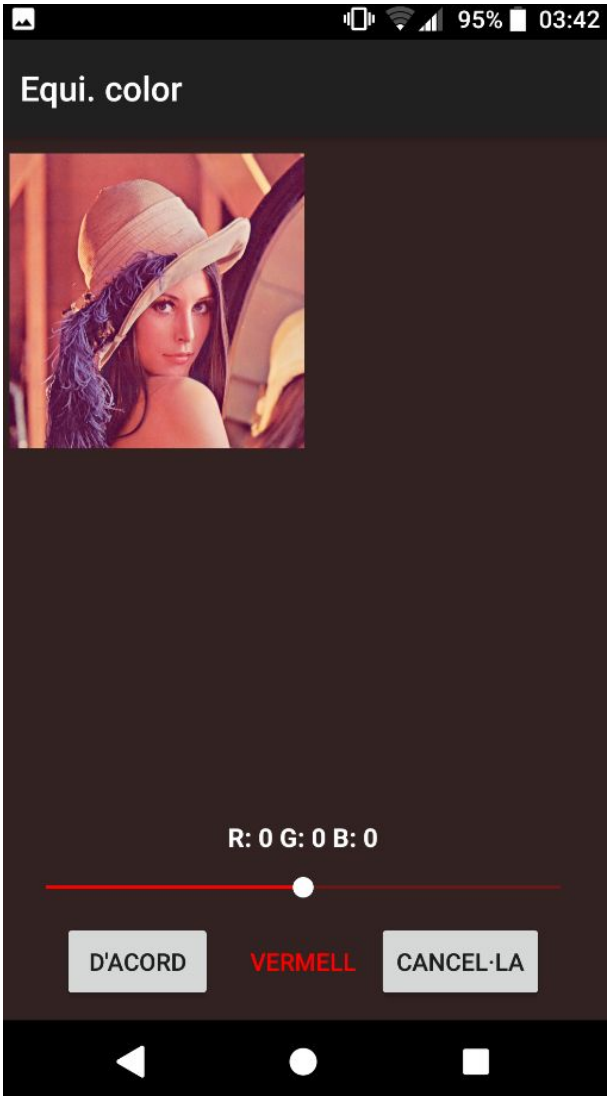
- Brillantor
- Contrast
- Balanç de blancs
- Zones il·luminades
- Zones d'ombra
- Nivell de blanc
- Nivell de negre
- Saturació
- Rotació d'angle
- Detall/Enfocament
- Opacitat



Captura de pantalla amb el layout per a efectes simples.

Efectes complexos (es controlen amb una sola SeekBar i un botó): tenen assignat el layout amb el valor GUIValue.SEEKBAR_COMPLEX:

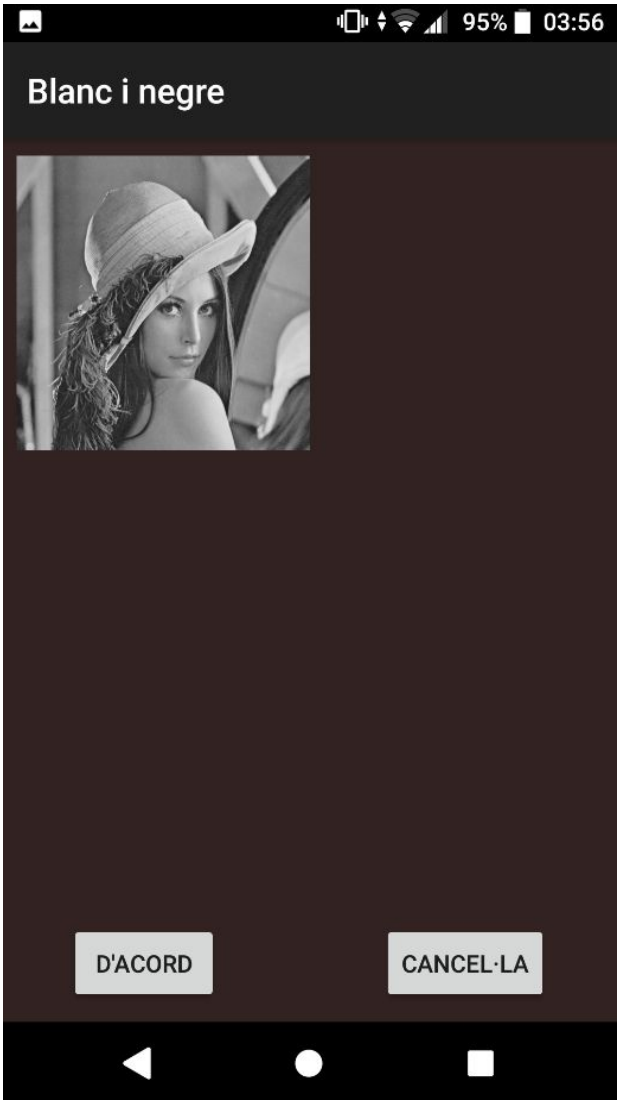
- Equilibri de color



Captura de pantalla amb el layout per a efectes complexos.

Efectes On/Off (es controlen amb un sol botó): tenen assignat el layout amb el valor GUIValue.ON_OFF:

- Invertir colors
- Blanc i negre
- Rotació 90º
- Rotació horitzontal
- Rotació vertical
- Retallar



Captura de pantalla amb el layout per a efectes on/off.

Classe TransmitValues

En la majoria d'efectes, el paràmetre d'entrada d'aplicació d'un efecte està format únicament per un sol valor enter, extret del valor d'una 'SeekBar' tal com s'ha comentat anteriorment, però hi ha dos casos en els quals l'efecte requereix més paràmetres. És el cas dels efectes "Equilibri de color" i "Retallar". Per solucionar aquest problema s'ha creat una classe amb certes propietats per a poder realitzar aquesta funció.

Equilibri de color

Aquest efecte es basa en l'edició dels components vermell, verd i blau per separat. És a dir, necessita tres valors diferents per a cadascun d'aquests components. Tenint en compte que amb la SeekBar únicament es pot transmetre un sol valor a la vegada, s'ha dissenyat un sistema de codificació de tres valors en un sol enter.

La classe disposa dels següents components:

Atributs:
- private static int valueR: valor d'edició del component vermell.
- private static int valueG: valor d'edició del component verd.
- private static int valueB: valor d'edició del component blau.

Mètodes:
- public static int getValueR(): retorna valueR.
- public static void setValueR(int valueR): assigna valueR.
- public static int getValueG(): retorna valueG.
- public static void setValueG(int valueG): assigna valueG.
- public static int getValueB(): retorna valueB.
- public static void setValueB(int valueB): assigna valueB.
- public static int getRGBCode(): es codifiquen en les tres posicions de menys pes de l'enter que es retornarà els valor que s'aplica sobre la component blava, en les tres de més pes els de la component vermella, i en les tres centrals els de la component verda.

  Exemple:

  R: 200 G: 100 B: 123 code = 200100123

- public static void setRGBCode(int code): descompon un enter segons la codificació anterior i n'assigna cada part a la variable que pertoca:

  Exemple:

  code = 150122050 R: 150 G: 122 B: 050

- public static boolean RGBisDefault(): retorna si els valors estan en el valor per defecte (és a dir, si no han estat modificats).
- public static void resetRGB(): assigna als tres valors al valor per defecte, que és 100.

S'ha de tenir en compte que en tot moment s'ha de controlar quina és la component de color que es vol modificar, i per això s'ha implementat un botó amb la funció exclusiva de controlar quin és el component a editar. El botó modifica el seu text i color de manera lineal: la primera vegada el seu estat està situat en la posició "Vermell", quan es clica canvia a "Verd", quan es torna a clicar canvia a "Blau" i així de manera seqüencial.
En cadascun d'aquests tres casos el que es modifica és quin mètode es crida a l'hora de passar el valor de la SeekBar. Quan està en la posició vermella es crida el mètode 'setValueR()', quan està en la posició verda es crida 'setValueG()' i quan està en la posició blava, es crida 'setValueB()'. Com a paràmetre d'entrada es passa el valor de la Seekbar.

D'aquesta manera es guarda quin és el valor a aplicar en l'efecte en els atributs de classe de TransmitValues, però això no és suficient, ja que l'efecte "Equilibri de color" els desconeix i els ha de poder obtenir d'alguna manera. És aquí quan entren en joc les funcions de codificació. Just després d'assignar un valor a qualsevol de les tres components, es fa una crida al mètode 'getRGBCode()', i aquest valor és el que s'assigna a l'efecte. Ara l'efecte ja conté el valor a aplicar, tot i estar codificat i no ser vàlid per a la seva aplicació.

Per poder llegir correctament el valor a aplicar a cada component, dins de l'efecte es realitza la crida habitual al mètode 'getBarValue()' que en aquest cas conté el valor codificat. Per tant, el que es fa a continuació, és assignar aquest valor codificat amb el mètode 'setRGBCode(code)' i seguidament s'obté cadascun dels tres valors necessaris amb els mètodes 'getValueR()', 'getValueG()' i 'getValueB()' respectivament. Una vegada s'ha obtingut aquest valor ja es pot processar la imatge de manera correcta.

Retallar

En aquest efecte es retalla el contingut de la imatge per tal de crear una versió més petita d'aquesta. Per tal d'indicar quina és la regió que es vol retallar, la lògica inicial ens fa pensar que es necessita un total de 4 paràmetres. Aquests són:

- Punt horitzontal inicial.
- Punt vertical inicial.
- Ample de l'àrea de retall.
- Altura de l'àrea de retall.

Recordem que, com en el cas anterior, els efectes guarden el seu valor d'aplicació en una variable de tipus int [35]. Segons la documentació oficial d'Android, aquest tipus de variable ens permet emmagatzemar com a màxim 32 bits. Això suposa que el valor més gran que pot emmagatzemar és el 2^31 - 1 = 2.147.483.647. Aquest valor es dedueix del fet següent: dels 32 bits que es poden guardar, la meitat són per a nombres positius i l'altra meitat per a nombres negatius. El zero es veu reflectit en la unitat que es resta a aquest valor. Per a poder guardar els valors comentats anteriorment, es pretén seguir el mateix esquema de codificació que en el cas de l'efecte Retallar. Suposem que es vol emmagatzemar simplement els valors en pixels. Es presenta el següent exemple:

Els valors 100 com a valor d'inici X, 200 d'inici Y, 300 d'ample i 400 d'alt es codificaran com a 100.200.300.400, que és un valor superior al valor màxim que pot emmagatzemar el tipus int. Això desencadena en un error de desbordament (overflow) sobre aquesta variable, el que fa que el nombre emmagatzemat no sigui el que realment es volia emmagatzemar. La següent opció que es decideix analitzar consisteix en el fet que, en lloc d'enviar els valors en pixels, es calcularà quin percentatge representa aquest pixel sobre la mida total de la imatge. De la mateixa manera, aplicarem aquesta conversió a l'ample i l'alt, així s'aconsegueix restringir el rang de valors possibles des de 0 fins a 100. Tot i això, encara es pot produir el mateix problema que en el cas anterior, ja que, per exemple, els percentatges d'inici molt probablement mai arribaran a 100%, però els d'ample i alt sí que és possible. Per tant, es podria produir el cas en què el valor a emmagatzemar sigui el 9.999.100.100, que també és major que 2.147.483.647.

Investigant una mica més s'arriba a la conclusió que un mètode que evita aquest problema és enviant els percentatges dels punts d'inici, la llargada de la diagonal en percentatge respecte a l'ample de la imatge, i l'angle d'aquesta respecte l'horitzontal.
El fet d'utilitzar l'angle i el percentatge de llargada de la diagonal limita que, tot i que el percentatge podria ser superior a 100, encara que normalment no acostuma a ser així, ja que les imatges normalment solen tenir una forma bastant rectangular, l'angle es limita a ser sempre inferior a 90. Amb aquesta combinació de valors, un possible valor màxim a emmagatzemar seria el 999.910.090, que és inferior al valor màxim a emmagatzemar sense patir error de desbordament.

Una vegada coneixent el funcionament bàsic d'aquest apartat de la classe TransmitValues, es descriuran els atributs i mètodes d'aquest:

Atributs:

- private static int cropCode: valor d'aplicació de l'efecte, codificat.
- private static int defaultCropCode: valor d'aplicació per defecte de l'efecte, codificat.

Mètodes:

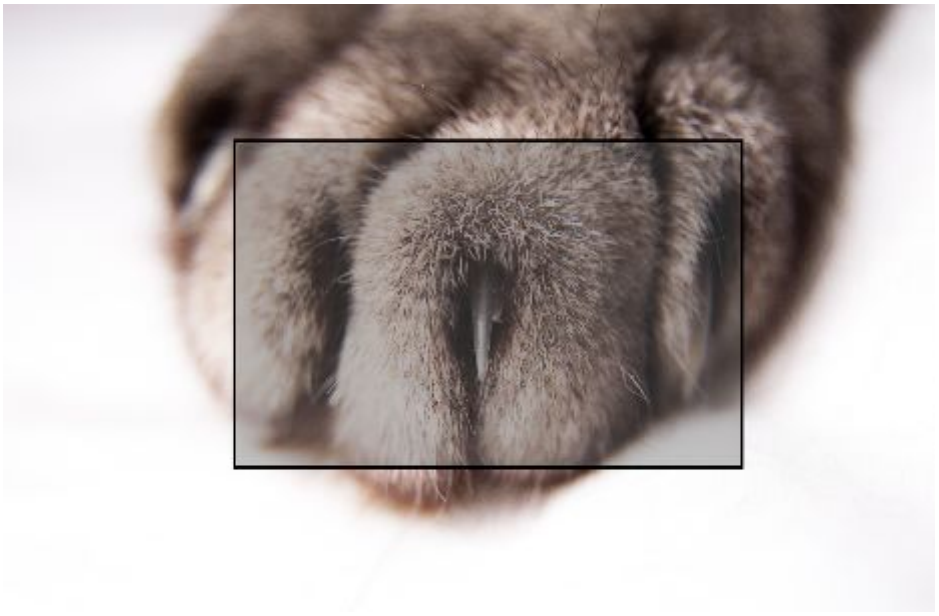
- public static int getCropCode(): retorna cropCode.
- public static void setCropCode(float x, float y, double diagonal, double angle): es codifiquen en les tres posicions de menys pes de l'enter que es retornarà el valor de l'angle. A les dues següents posicions de menys pes, el percentatge de grandària de la diagonal respecte l'amplada de la imatge, en les dues següents posicions, el percentatge d'inici vertical, i en les dues de més pes, el percentatge d'inici horitzontal.

Exemple:

- x: 76
- y: 82
- diagonal: 120
- angle: 35
- cropCode = 768212035

- public static void setCropCode(int i): assigna a cropCode un valor.
- public static boolean CROPIsDefault(): retorna si el valor d'aplicació de l'efecte cropCode és igual al valor per defecte de l'efecte.
- public static void setDefaultCropCode(int defaultCropCode): assigna un valor per defecte a defaultCropCode.

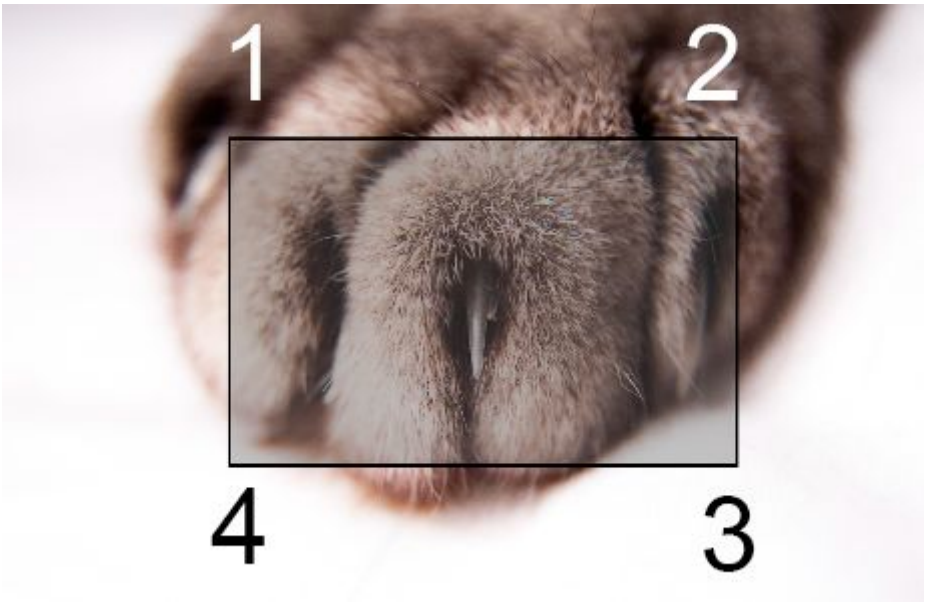
Per a poder decidir quina és l'àrea de retall de la imatge, s'ha implementat una combinació d'elements de la classe View per tal de poder facilitar aquesta tasca a l'usuari. S'ha implementat un total de 4 View de color negre que realitzen la funció de límits de retall, juntament amb un altre View, el qual és de color gris amb una opacitat del 50% per a remarcar l'àrea seleccionada.



Àrea de retall.

Per a una correcta implementació d'aquest efecte s'ha creat les següents funcions:

- private boolean moveAreaOrPoint(MotionEvent event): donat un event de clic, es determina segons la regió de la pantalla que s'ha clicat, si s'ha de modificar alguna de les quatre cantonades de l'àrea de retall, és a dir, modificar la mida de l'àrea. En aquest cas retorna 'false'. En cas contrari, retorna true, i per tant, significa que s'ha de desplaçar el quadrat per tal de modificar l'àrea de retall.
- private int getPointToMove(MotionEvent event): donat un event de clic, es calcula quina de les 4 cantonades de l'àrea de retall és la més pròxima a la regió de la pantalla que s'ha fet clic. El retorn d'aquesta funció es correspon segons l'esquema següent:



Valor de retorn assignat a cada cantonada.

En la implementació del mètode 'onTouch()' de MainActivity és on es du a terme tot el procés de modificació de l'àrea de retall, ja que la totalitat de la configuració d'aquest efecte es realitza mitjançant la interacció tàctil amb l'usuari.
Primerament es comprova si s'està aplicant l'efecte Retallar. En cas afirmatiu, es canvia la visibilitat dels View a VISIBLE, perquè l'usuari pugui interactuar amb ells de forma correcta. Tot seguit, si el tipus d'event es correspon a MotionEvent.ACTION_DOWN, és a dir, la part del clic on l'usuari toca la pantalla per primer cop, es comprova si s'ha de moure l'àrea o una de les quatre cantonades amb la funció 'moveAreaOrPoint()'. Si es mou l'àrea, es calcula la distància del clic amb la cantonada superior esquerra, per tal que el moviment del requadre de retall sigui el correcte. En el cas contrari, si es mou un punt, es calcula quin punt es mou amb la funció 'getPointToMove()'. Es repeteix l'opció del càlcul de distàncies per poder realitzar un correcte moviment de la cantonada que es veu afectada.

Després de calcular les accions a realitzar, es procedeix a modificar l'àrea de retall o la cantonada corresponent. En el cas de l'àrea, es comprova que l'àrea de retall no pugui sortir de la imatge original. En el cas dels punts, es du a terme la mateixa comprovació, a més a més del fet que els punts no se superposin entre ells.

Finalment, una vegada s'ha realitzat tots els càlculs de la posició on s'ha de mostrar cada View, es procedeix a canviar les mides d'amplada i alçada perquè aquestes siguin les correctes. Una vegada fet això, es continua amb el càlcul dels valors necessaris per enviar a través de la classe TransmitValues.
Els valors de percentatge d'inici horitzontal i vertical s'obtenen dividint el valor en pixels entre l'amplada i l'alçada respectivament. Després, es multiplica per 100 per a obtenir un valor dins del rang 0-100.
La diagonal s'obté aplicant el teorema de Pitàgores, i dividint aquesta distància entre l'amplada de la imatge i multiplicant-la per 100 pel motiu explicat anteriorment.
L'angle s'obté a través de l'arctangent de la relació entre l'alt i l'ample de l'àrea de retall. Després, es converteix en graus, ja que aquest resultat està en radiants. S'utilitza la llibreria Math [36] per a realitzar aquests càlculs. Finalment s'emmagatzemen aquests valors amb el mètode 'setCropCode()' de TransmitValues, i cridant a 'getCropCode()' obtindrem el valor codificat i llest per a poder emmagatzemar a l'efecte amb el mètode 'setBarValue()'.

Una vegada dins de l'execució del codi de l'efecte, el primer que es fa és agafar el valor que hi ha emmagatzemat en el valor d'aplicació, que s'obté mitjançant el mètode 'getBarValue()'. Seguidament es procedeix a la descodificació d'aquest valor. Per fer-ho, es descompon el nombre en les parts que s'ha explicat anteriorment a través dels operadors divisió i mòdul. Després es transformen els percentatges d'inici en pixels, i es prossegueix fent el mateix amb la diagonal i l'angle, per a obtenir l'alt i l'ample en pixels de l'àrea a retallar.
S'ha de tenir en compte que durant tot aquest procés s'ha eliminat els decimals dels valors que s'emmagatzemaven, fet que pot provocar un error en la reconversió d'aquests valors. És per això que es fa un petit control per tal de comprovar que les mides recuperades no superin els límits de la imatge, ja que desencadena en errors greus.
Una vegada realitzats tots aquests passos, s'obté el nou Bitmap utilitzant les variables descodificades i la funció 'getPixels()' sobre el Bitmap original, passant com a paràmetres aquests valors.

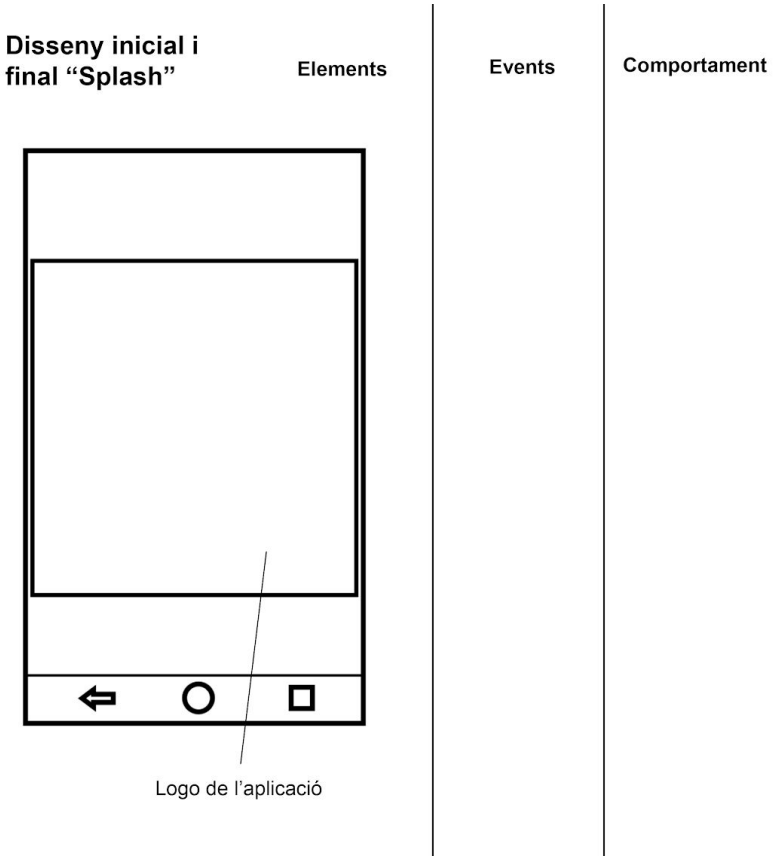
5 Resultats

Una vegada finalitzades les etapes d'anàlisi, disseny i desenvolupament, es procedeix a l'exposició dels resultats que s'ha obtingut. El resultat ha estat que s'ha pogut realitzar una app per a editar imatges amb bastants dels elements que es tenia pensat inicialment i alguns dels dissenys inicials han patit modificacions, com es mostra a continuació.
Tot i que s'ha intentat implementar mètodes per tal d'optimitzar el nombre de càlculs necessaris en cadascun dels efectes, en alguns casos això no ha estat suficient, ja que en alguns dispositius que no són de gamma alta i que no incorporen processadors de gran potència, a l'hora d'aplicar alguns efectes, s'observa l'aplicació no és tan fluida com es desitjaria que fos. Tot i això, en general el resultat obtingut es qualifica com a l'esperat.

5.1 Aplicació final

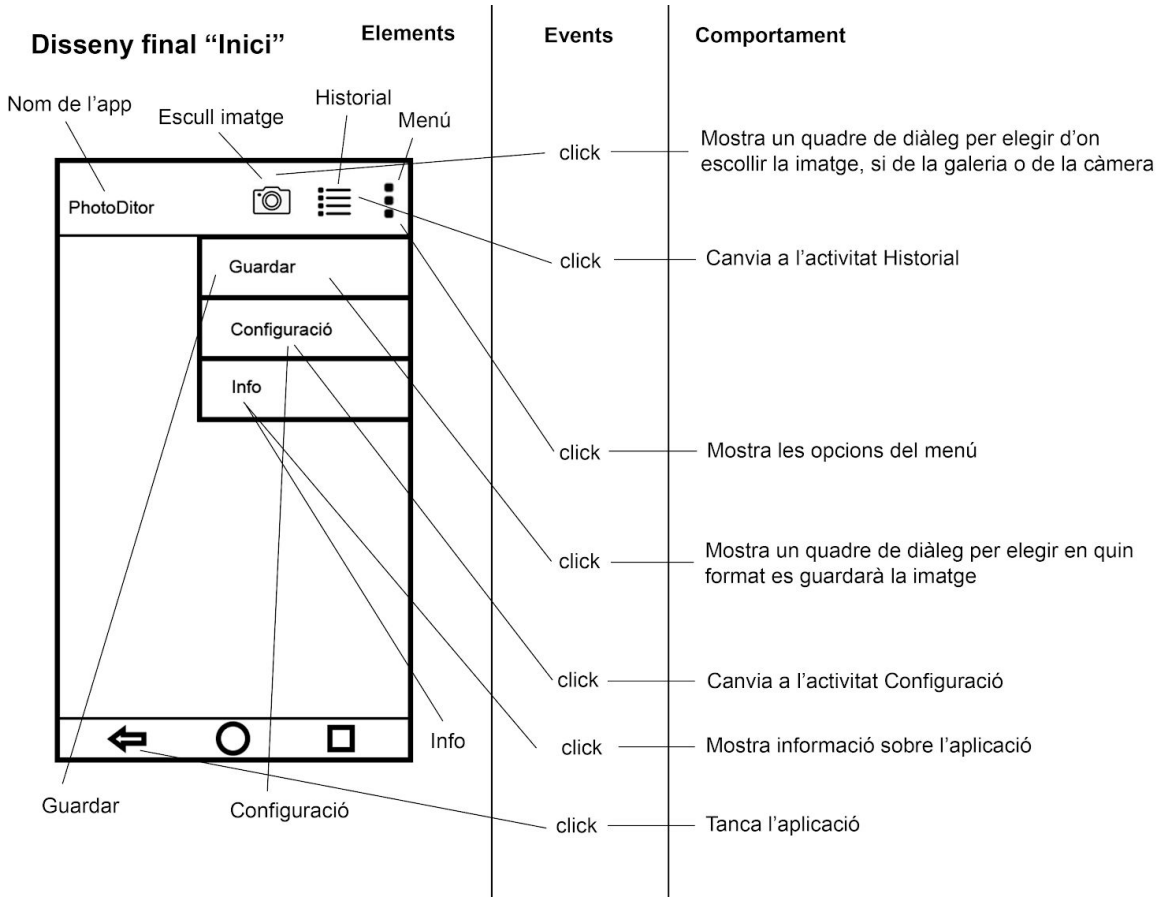
Durant l'evolució del projecte els dissenys inicials han patit modificacions, que en alguns casos han estat millores i en altres optimitzacions a causa de l'espai disponible en pantalla. Els dissenys finals de cada pantalla es descriuen a continuació:

Dissenys final pantalla 1 (Splash Screen)

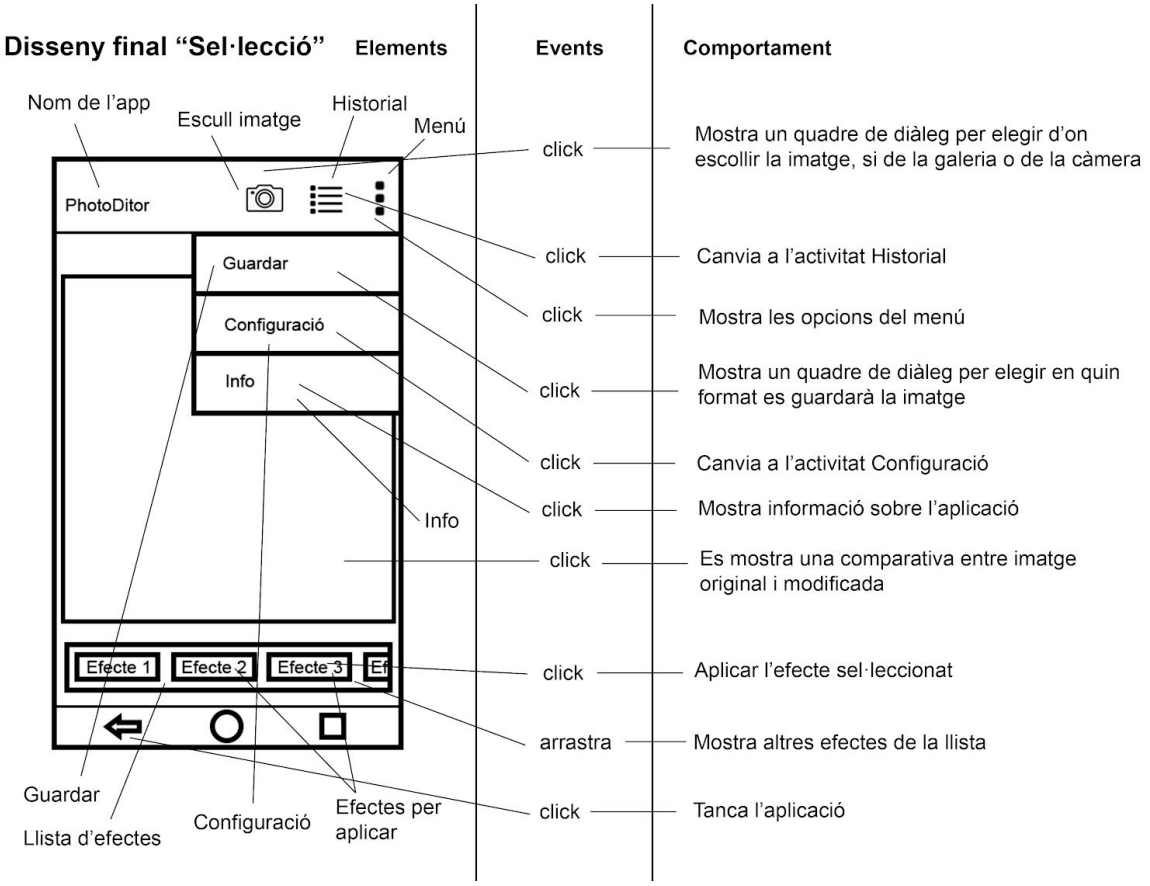


Dissenys final pantalla 2 (Aplicació d'efectes)

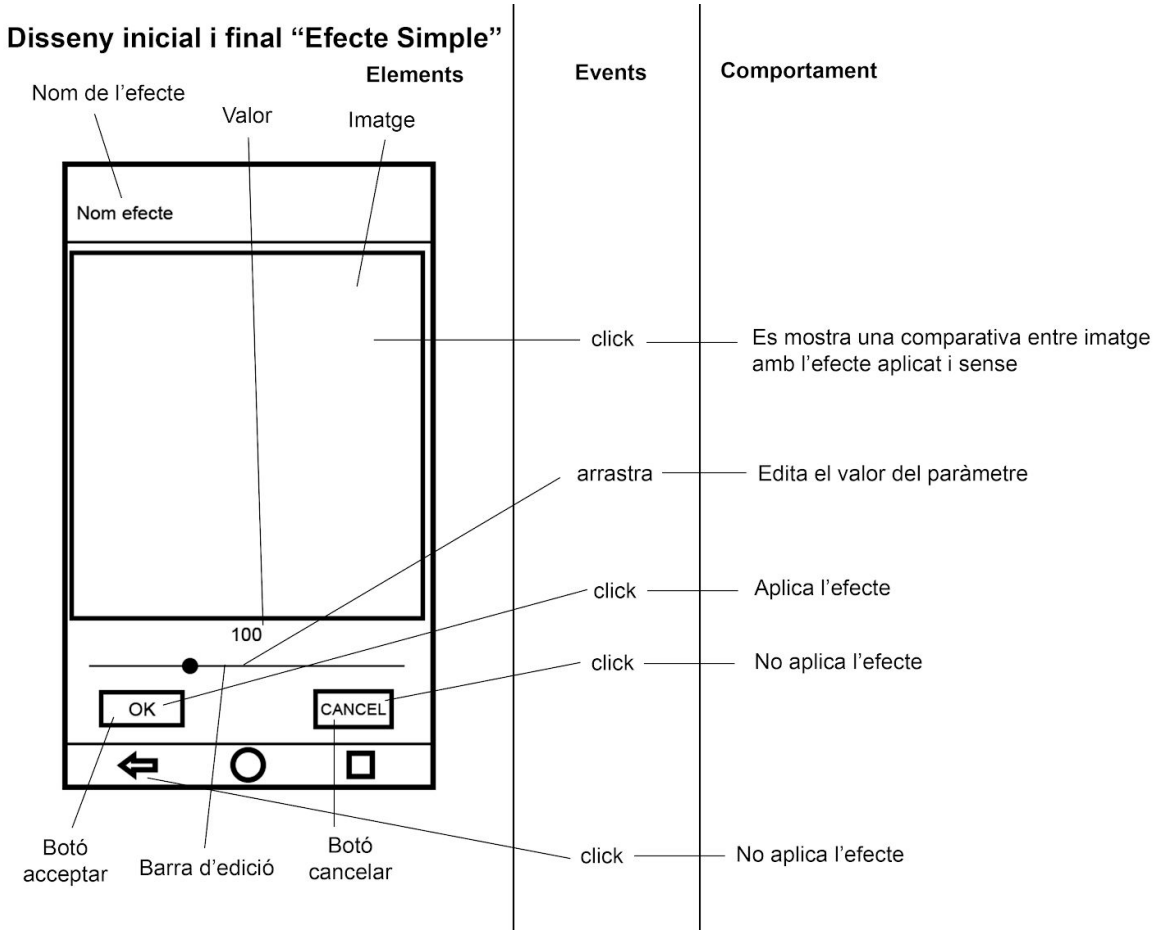
Pantalla inicial



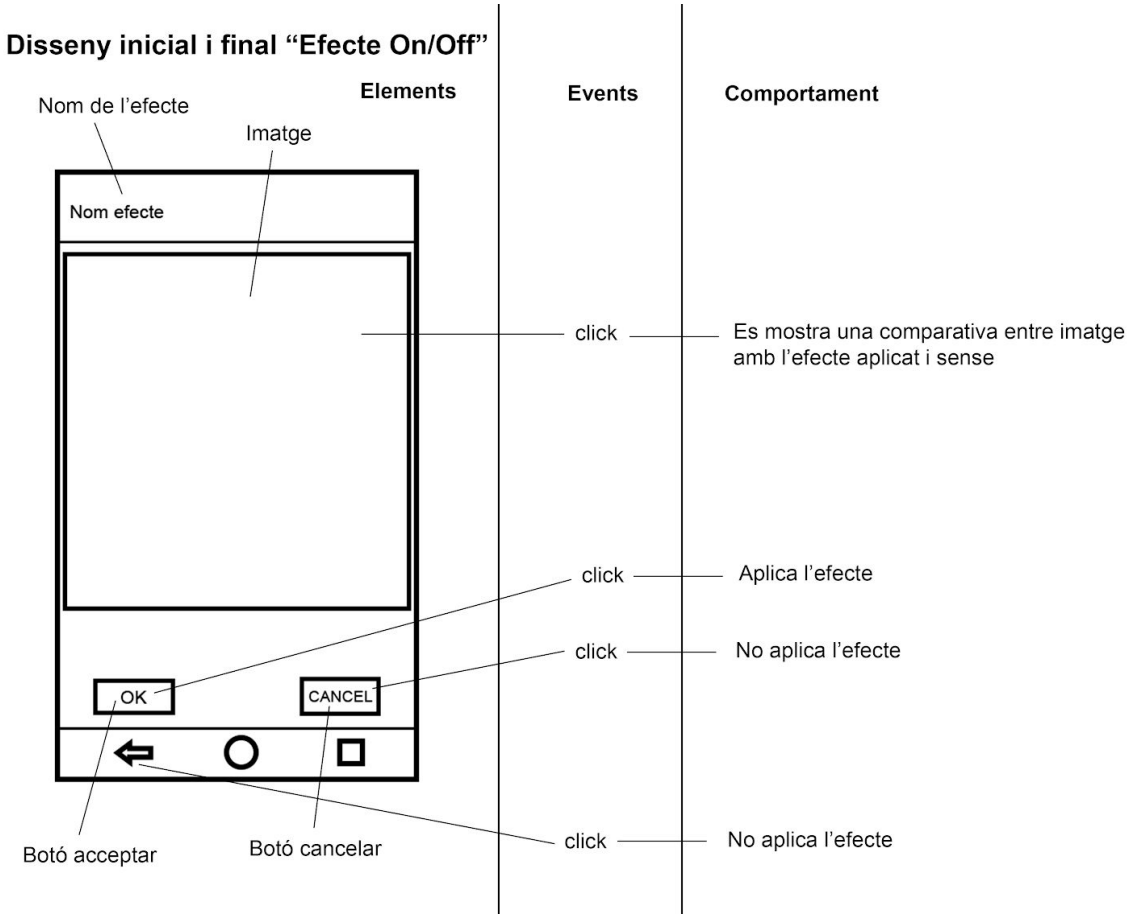
Seleccionar efecte



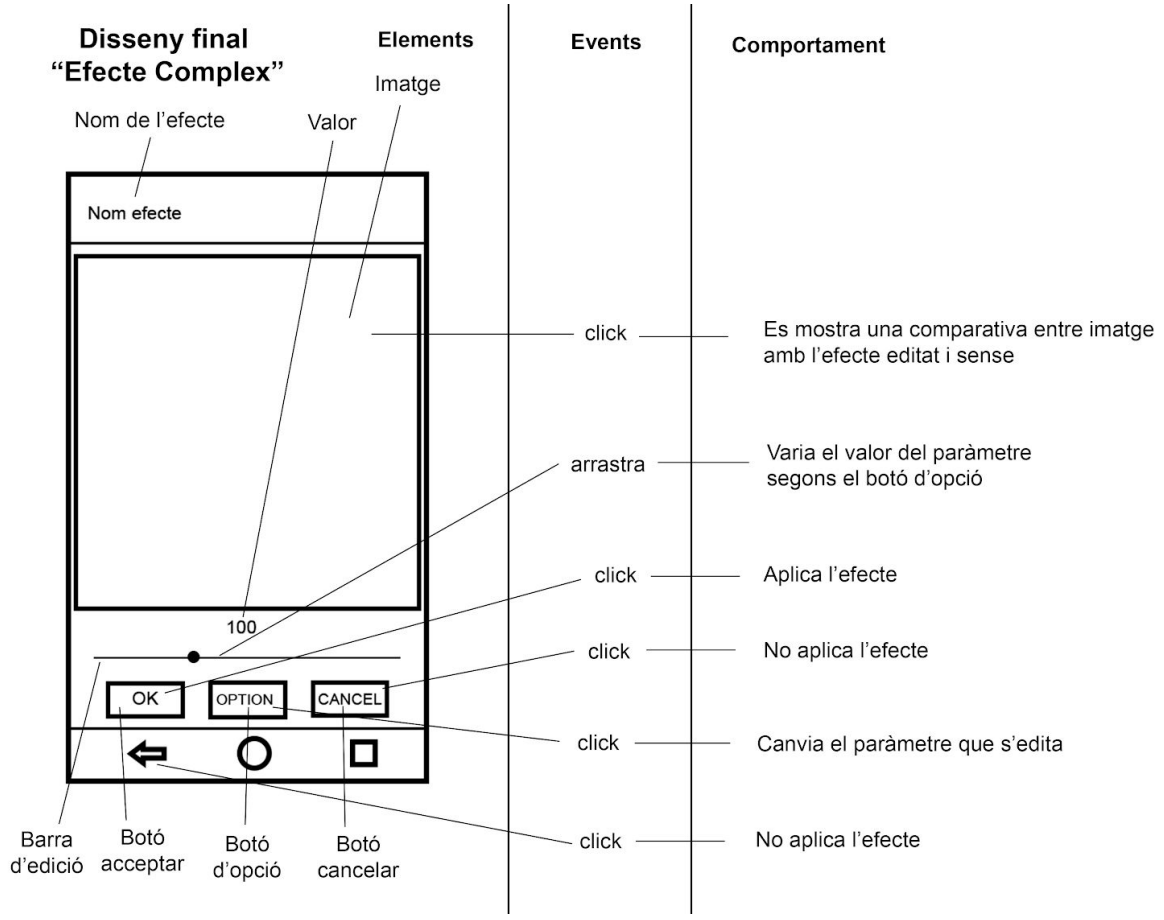
Efecte simple



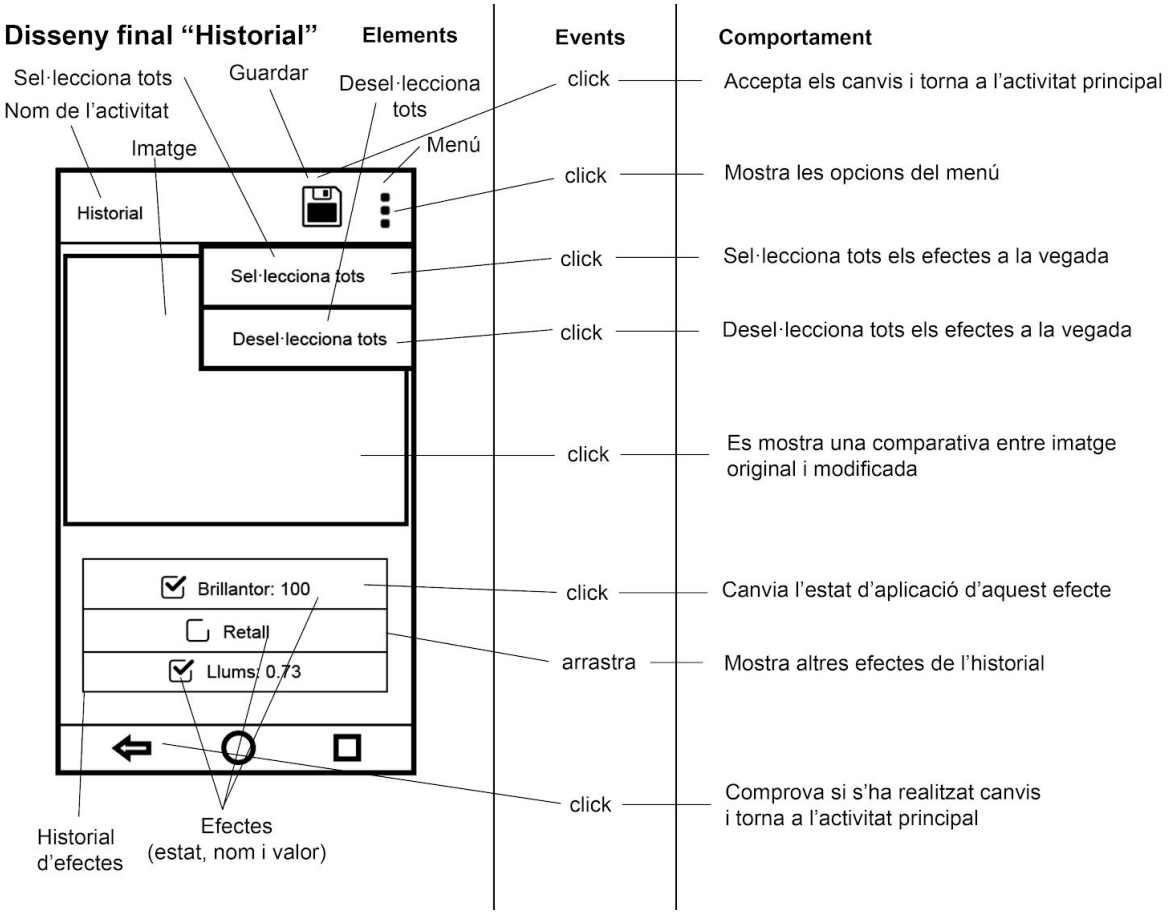
Efecte On/Off



Efecte complex



Disseny final pantalla 3 (Historial)



La duració del desenvolupament d'aquest projecte ha estat aproximadament d'uns 7 mesos, dels quals se'n desglossa 6 per al desenvolupament de l'aplicació i 1 per la redacció de la memòria, tot i que aquestes dues tasques s'han desenvolupat paral·lelament.
En el desenvolupament s'inclou l'estudi de mercat, el disseny de l'aplicació, la implementació d'aquesta i la resolució de problemes. Durant els 6 mesos de desenvolupament s'estima que s'ha treballat aproximadament unes 5 hores diàries els dies laborals, fet que resulta en un total de 600 hores.

En termes d'arxius, s'ha generat un total de 1.838 arxius, tot i que realment la gran majoria d'aquests han estat creats per defecte per Android Studio, ja que són tant arxius necessaris per al compilador Gradle com propis de l'aplicació però que no s'han modificat. Fent referència al nombre d'arxius que s'ha modificat, es comptabilitza un total de 48 arxius (manifest, classes, activitats, arxius xml i altres), que es tradueix en un total de 276.465 línies de codi total tenint en compte tots els arxius de l'aplicació.

Aquestes dades s'han obtingut mitjançant el plug-in 'Statistic' [37] d'Android Studio, ignorant en els resultats els arxius que crea software Git.

Per descarregar l'aplicació a través de Google Play utilitzeu el següent enllaç:

https://play.google.com/store/apps/details?id=kikevite.photoditor

El codi de l'aplicació està disponible de forma pública al website GitHub, en el qual podeu accedir a través del següent enllaç:

https://github.com/kikevite/EfectesJunts

5.2 Pressupost

S'ha calculat que el pressupost final d'aquesta aplicació es correspon al següent:

Com s'ha comentat anteriorment, en el desenvolupament d'aquesta aplicació s'ha estimat un total d'unes 600 hores per al desenvolupament. Es tindrà en compte un salari net d'aproximadament 9€/ hora, extret d'experiències laborals realitzades anteriorment dins de l'àmbit de la programació i semblants al desenvolupament d'aplicacions.

En termes de equipament informàtic per al desenvolupament s'ha comptabilitzat un ordinador portàtil, una pantalla externa i un ratolí. Tot i que per a les proves de l'aplicació s'ha utilitzat un smartphone personal, aquest queda exclòs del pressupost ja que no és estrictament necessari. Android Studio proporciona una eina de creació de dispositius virtuals que pot ser utilitzada com a tal.

Concepte	Quantitat	Preu	Subtotal
Hores de treball	600 hores	9€	5.400€
Ordinador	1	700€	700€
Pantalla Externa	1	100€	100€
Mouse	1	20€	20€
Publicació Play Store	1	22€	22€
		Total	**6.242€**

6 Conclusions

Una vegada ha finalitzat el desenvolupament del projecte, es procedeix a analitzar els resultats obtinguts per tal d'extreure conclusions sobre aquests: a nivell objectiu, si s'ha aconseguit complir els objectius marcats, si les idees inicials han patit modificacions, si s'ha aplicat un desenvolupament correcte. A nivell subjectiu, si s'ha treballat d'una manera còmoda, eficient i si els resultats finals han complert les expectatives.

Una vegada s'ha finalitzat aquest projecte, es procedeix a analitzar els resultats que s'ha obtingut. Es compara els objectius inicials amb aquests resultats per a saber quins objectius s'ha complert.

Recordem els objectius inicials més importants del projecte i quin ha estat el resultat final:

- Crear una app per a editar fotos: s'ha aconseguit crear una app amb un bon nombre total d'efectes tot i no haver-se pogut implementar tots a causa de la complicació a nivell de programació que alguns d'ells comporta. A part, també s'ha aconseguit que l'aplicació compleixi uns certs requisits quant a disseny, col·locant cadascun dels diferents elements en les zones que resulta més accessible per a la seva utilització.

- Publicar l'app a Play Store: l'aplicació es troba disponible a la Play Store des del 22 de setembre de 2018 i s'ha actualitzat unes quantes vegades per tal de corregir errors. Per tant, s'ha aconseguit un altre dels objectius que s'havia establert.

- Investigar algoritmes de processament d'imatge: tot i que d'alguns algoritmes de processament ja se'n coneixia el funcionament, se n'han descobert de nous. També s'ha après a implementar els efectes i a treballar amb imatges en llenguatge Java, utilitzant tant algunes classes ja creades com creant-ne de noves.

- Millorar utilització d'Android Studio: s'ha millorat bastant el nivell de coneixement sobre la utilització dels elements que formen part d'Android Studio, com per exemple el Profiler, utilitzat en l'apartat d'implementació dels efectes per tal d'optimitzar els temps de càlcul de cadascun d'aquests. També s'ha après a utilitzar el debugador, que en alguns moments ha estat essencial per a poder corregir errors importants.

- Ampliar coneixements sobre Java: a part d'haver après a tractar imatges en Java, també s'ha après alguns aspectes que es desconeixien, com per exemple classes noves o patrons de disseny, com el "Command Pattern".

6.1 Treball futur

En versions futures d'aquesta app es podria implementar un nombre major d'efectes i eines que permetin a un fotògraf facilitar alguns càlculs i consultar algunes dades generals, com podrien ser les següents:

- Versió per a iPhone i Windows Phone:

Es podria fer una versió d'iPhone i Windows Phone amb el software de Xamarin per exemple. És una plataforma en la qual només s'utilitza un sol llenguatge de programació i ho "tradueix" per a fer les versions corresponents.

- Calculadora de profunditat de camp:

Indica al fotògraf quina zona de la imatge sortirà correctament enfocada introduint les dades de la configuració actual de la càmera, o a la inversa. Útil sobretot en fotografia de paisatges on s'aconsegueix la profunditat de camp més gran quan s'enfoca a l'anomenada distància hiperfocal.

- Golden Hour:

Es mostra quines són les hores del dia que es corresponen a les "hores daurades", moments aproximadament corresponents a la sortida i posta del Sol en els quals la llum que emet aquest adquireix unes qualitats molt atractives de cara a la fotografia.

- Visualització del temps (meteorologia):

Un simple visualitzador meteorològic per saber si en una zona determinada hi ha núvols o mal temps, per evitar viatges innecessaris.

- Disparador i configuració de càmeres:

Control remot universal de la càmera fotogràfica des d'un smartphone a través de Bluetooth o Wi-Fi per tal d'evitar que cada vegada que l'usuari canviï de càmera s'hagi de comprar un disparador nou.

- Transmissió d'imatges de la càmera al mòbil:

Enviar les imatges capturades amb la càmera fotogràfica al smartphone a través de Bluetooth o Wi-Fi, per a poder compartir-les a les xarxes socials o editar-les i guardar-les.

6.2 Reflexions finals

Des de l'anàlisi dels resultats obtinguts, en termes subjectius, el desenvolupament d'una aplicació resulta bastant complex. Això és degut a que algunes situacions comporten una càrrega de feina bastant elevada. Per tal d'evitar aquests problemes i situacions semblant, es realitza una reflexió a nivell personal sobre aspectes a millorar en futurs projectes.

De forma general s'ha observat que en molts casos les idees inicials han variat molt, fins al punt en que l'aplicació ha patit modificacions a nivell d'estructura de dades, que segons en quina fase del desenvolupament pot resultar molt complex de realitzar, per no dir que és bastant arriscat. Per a corregir aquest error es creu oportú analitzar encara amb més profunditat com es vol implementar les idees que es volen realitzar des de bon principi. En moltes ocasions les ganes de posar-se a fer feina i d'obtenir resultats de manera ràpida sobrepassa el nivell d'anàlisi que s'hauria de realitzar abans de començar. Aquest és un comportament que en futurs projectes es tindrà en compte i que servirà per tal d'estalviar molt de temps que es desaprofita modificant conceptes que es podria haver implementat d'una forma millor des del principi.

Un dels aspectes què tot i haver intentat aplicar de forma correcta no s'ha aconseguit d'una manera massa eficient, és el d'escriure el codi de manera organitzada, clara i comentant cadascuna de les parts del projecte que pot ser que no s'entenguin a primera vista. Aquest aspecte és important perquè durant el desenvolupament de l'app el propi creador del codi pot no recordar amb exactitud quines funcions realitza cada mètode, i el més important, pot ser que altres persones s'afegeixin al projecte posteriorment i que no entenguin certes parts del codi. En aquest aspecte es podria considerar que això no ha resultat un problema ja que desde bon inici es tenia present el fet de que aquest projecte es realitzaria de forma individual. Tot i que en la majoria del possible s'ha intentat comentar el codi, en algunes ocasions això no s'ha fet.

Pel que fa a opinió final del projecte, aquesta és positiva, i el més important, es té la sensació d'haver creat una aplicació que tot i no ser la millor del mercat, continuaré utilitzant en un futur. El descobriment i aprenentatge de nous paradigmes de programació m'ha resultat atractiu fins al punt que podria ser una possibilitat buscar futures oportunitats de treball que estiguin encarades en el desenvolupament d'aplicacions.

7 Referències

[1] Image Processing Algorithms Part 3: Greyscale Conversion – Dreamland Fantasy Studios. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-3-greyscale-conversion/.

[2] Image Processing Algorithms Part 4: Brightness Adjustment – Dreamland Fantasy Studios. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-4-brightness-adjustment/.

[3] image processing - How to programmatically change contrast of a bitmap in android? - Stack Overflow. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://stackoverflow.com/questions/12891520/how-to-programmatically-change-contrast-of-a-bitmap-in-android.

[4] How to Blur Images Efficiently with Android's RenderScript. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://futurestud.io/tutorials/how-to-blur-images-efficiently-with-androids-renderscript.

[5] Image Processing Algorithms Part 7: Colour Inversion And Solarisation – Dreamland Fantasy Studios. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://www.dfstudios.co.uk/articles/programming/image-programming-algorithms/image-processing-algorithms-part-7-colour-inversion-solarisation/.

[5] Android Rotate and Scale Bitmap Example | More Is Not Always Better. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://blahti.wordpress.com/2014/02/03/android-rotate-scale-bitmap/.

[6] java - Flip a Bitmap image horizontally or vertically - Stack Overflow. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://stackoverflow.com/questions/36493977/flip-a-bitmap-image-horizontally-or-vertically.

[7] Actividades | Android Developers. A: [en línia]. [Consulta: 7 octubre 2018]. Disponible a: https://developer.android.com/guide/components/activities?hl=es-419.

[8] Manifiesto de la app | Android Developers. A: [en línia]. [Consulta: 7 octubre 2018]. Disponible a: https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419.

[9] Buttons | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/guide/topics/ui/controls/button.

[10] SeekBar | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/widget/SeekBar.

[11] ImageView | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/widget/ImageView.

[12] Bitmap | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/graphics/Bitmap.

[13] View.OnTouchListener | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/view/View.OnTouchListener?hl=Es-419.

[14] Create a List with RecyclerView | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/guide/topics/ui/layout/recyclerview.

[15]. RecyclerView.Adapter | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/support/v7/widget/RecyclerView.Adapter.

[16] DividerItemDecoration | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/support/v7/widget/DividerItemDecoration.

[17] Cuadros de diálogo | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/guide/topics/ui/dialogs.

[18] AlertDialog.Builder | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/support/v7/app/AlertDialog.Builder.

[19] Toasts overview | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/guide/topics/ui/notifiers/toasts.

[20] Gravity | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/view/Gravity.

[21] Menús | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/guide/topics/ui/menus.

[22] TextView | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/widget/TextView.

[23] Cómo solicitar permisos durante el tiempo de ejecución | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/training/permissions/requesting?hl=es-419.

[24] Intents y filtros de intents | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/guide/components/intents-filters.

[25] Reveal or hide a view using animation | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/training/animation/reveal-or-hide-view.

[26] Crear íconos de apps con Image Asset Studio | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/studio/write/image-asset-studio.

[27] FileProvider | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/support/v4/content/FileProvider.

[28] Calendar | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/java/util/Calendar.

[29] AsyncTask | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/os/AsyncTask.

[30] Contributors, W.Command pattern. A: 24 September 2018 00:40 UTC [en línia]. [Consulta: 19 octubre 2018]. Disponible a: https://en.wikipedia.org/wiki/Command_pattern.

[31] RenderScript Overview | Android Developers. A: [en línia]. [Consulta: 1 octubre 2018]. Disponible a: https://developer.android.com/guide/topics/renderscript/compute.

[32] ScriptIntrinsicBlur | Android Developers. A: [en línia]. [Consulta: 1 octubre 2018]. Disponible a: https://developer.android.com/reference/android/renderscript/ScriptIntrinsicBlur.

[33] Allocation | Android Developers. A: [en línia]. [Consulta: 1 octubre 2018]. Disponible a: https://developer.android.com/reference/android/renderscript/Allocation.

[34] Matrix | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/android/graphics/Matrix.

[35] Integer | Android Developers. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://developer.android.com/reference/java/lang/Integer.

[36] Math | Android Developers. A: [en línia]. [Consulta: 2 octubre 2018]. Disponible a: https://developer.android.com/reference/java/lang/Math.

[37] Statistic - Plugins | JetBrains. A: [en línia]. [Consulta: 8 octubre 2018]. Disponible a: https://plugins.jetbrains.com/plugin/4509-statistic.

8 Bibliografia

Gamma, E., 1995. Design patterns : elements of reusable object-oriented software. Addison-Wesley. ISBN 0201633612.

Grau, A. autor., 2017. Digital image processing. Basic functions / Antoni Grau, Xie Jin [en línia]. [Barcelona] : ESAII,. [Consulta: 5 octubre 2018]. Disponible a: https://discovery.upc.edu/iii/encore/record/C__Rb1489800?lang=cat.

Oliva, D. i Cuevas, E., 2017. Advances and Applications of Optimised Algorithms in Image Processing [en línia]. Cham: Springer International Publishing. Intelligent Systems Reference Library. ISBN 978-3-319-48549-2. DOI 10.1007/978-3-319-48550-8. [Consulta: 5 octubre 2018]. Disponible a: http://link.springer.com/10.1007/978-3-319-48550-8.

Aspectos fundamentales de la aplicación | Android Developers. A: [en línia]. [Consulta: 7 octubre 2018]. Disponible a: https://developer.android.com/guide/components/fundamentals?hl=es-419

Photoshop User Guide. A: [en línia]. [Consulta: 26 setembre 2018]. Disponible a: https://helpx.adobe.com/es/photoshop/user-guide.html?topic=/es/es/photoshop/morehelp/image_adjustments.ug.js.